MySQL 8.0 Reference Manual Including MySQL NDB Cluster 8.0

|  |  |  |
| --- | --- | --- |
| author | liangdu |  |
| email | [liangdu1992@gmail.com](mailto:liangdu1992@gmail.com) |  |
| version | v1.0 |  |
| date | 2021-04-21 |  |

**摘要**

这是MySQL™参考手册。它分别记录了MySQL 8.0至8.0.25，以及基于NDB 8.0至8.0.23-ndb-8.0.23版本的NDB Cluster版本。它可能包括尚未发布的MySQL版本的功能文档。有关哪些版本已经发布的信息，请参见MySQL 8.0发布说明。

MySQL 8.0的特性。本手册描述的功能并不包括在MySQL 8.0的每个版本中；这些功能可能不包括在授权给你的MySQL 8.0版本中。如果你对包括在你的MySQL 8.0版本中的功能有任何疑问，请参阅你的MySQL 8.0许可协议或联系你的Oracle销售代表。

有关详细说明每个版本中的变化的注释，请参见MySQL 8.0版本注释。

有关法律信息，包括许可信息，见前言和法律声明。

有关使用MySQL的帮助，请访问MySQL论坛，在那里你可以与其他MySQL用户讨论你的问题。

文件产生于：2021-01-20（修订：68545）。

通过www.DeepL.com/Translator（免费版）翻译

**前言和法律声明**

本手册是MySQL数据库系统8.0版的参考手册，直至8.0.25版。MySQL 8.0各次要版本之间的差异在本文本中以发布号（8.0.x）为参考进行说明。关于许可证信息，请参见法律声明。

由于MySQL 8.0与以前的版本在功能和其他方面存在许多差异，本手册不打算用于旧版本的MySQL软件。如果你使用的是MySQL软件的早期版本，请参考相应的手册。例如，MySQL 5.7参考手册涵盖了5.7系列的MySQL软件版本。

许可信息-MySQL 8.0。本产品可能包括第三方软件，在许可证下使用。如果你使用的是MySQL 8.0的商业版本，请参阅《MySQL 8.0商业版本许可信息用户手册》以了解许可信息，包括与可能包含在该商业版本中的第三方软件有关的许可信息。如果你使用的是MySQL 8.0的社区版，请参阅《MySQL 8.0社区版许可信息用户手册》以了解许可信息，包括与可能包含在该社区版中的第三方软件有关的许可信息。

许可信息-MySQL NDB Cluster 8.0。如果你使用的是MySQL NDB Cluster 8.0的社区版本，请参阅《MySQL NDB Cluster 8.0社区版本许可信息用户手册》了解许可信息，包括与可能包含在该社区版本中的第三方软件有关的许可信息。

**法律声明**

Copyright © 1997, 2021, Oracle和/或其附属机构。

本软件和相关文档是根据包含使用和披露限制的许可协议提供的，受到知识产权法的保护。除非你的许可协议明确允许或法律允许，否则你不得以任何形式或手段使用、拷贝、复制、翻译、广播、修改、许可、传输、分发、展览、执行、出版或展示任何部分。禁止对本软件进行逆向工程、反汇编或反编译，除非法律要求具有互操作性。

此处包含的信息如有变化，恕不另行通知，也不保证没有错误。如果你发现任何错误，请以书面形式向我们报告。

如果这是交付给美国政府或代表美国政府授权的任何人的软件或相关文件，那么以下通知是适用的。

美国政府终端用户。甲骨文程序（包括任何操作系统、集成软件、任何嵌入、安装或激活在交付的硬件上的程序，以及对这些程序的修改）和甲骨文计算机文档或其他交付给美国政府最终用户或由其访问的甲骨文数据，根据适用的联邦采购条例和机构特定的补充条例，属于 "商业计算机软件 "或 "商业计算机软件文档"。因此，对i)甲骨文程序（包括任何操作系统、集成软件、任何嵌入、安装或激活在交付的硬件上的程序以及这些程序的修改）、ii)甲骨文计算机文档和/或iii)其他甲骨文数据的使用、复制、发行、展示、披露、修改、准备衍生作品和/或改编，都要遵守适用合同中所载的许可规定的权利和限制。管理美国政府使用甲骨文云服务的条款由此类服务的适用合同规定。没有授予美国政府任何其他权利。

该软件或硬件是为在各种信息管理应用中的一般用途而开发的。它不是为任何固有的危险应用而开发或打算使用的，包括可能产生人身伤害风险的应用。如果你在危险的应用中使用本软件或硬件，那么你应负责采取所有适当的防故障、备份、冗余和其他措施来确保其安全使用。甲骨文公司及其附属机构对在危险应用中使用本软件或硬件造成的任何损失不承担任何责任。

Oracle和Java是Oracle和/或其附属机构的注册商标。其他名称可能是其各自所有者的商标。

Intel和Intel Inside是Intel公司的商标或注册商标。所有SPARC商标都是在授权下使用的，是SPARC国际公司的商标或注册商标。AMD、Epyc和AMD标志是Advanced Micro Devices的商标或注册商标。UNIX是The Open Group的注册商标。

本软件或硬件和文档可能提供对第三方内容、产品和服务的访问或相关信息。甲骨文公司及其附属机构对第三方的内容、产品和服务不负责任，并明确拒绝提供任何形式的保证，除非您与甲骨文公司之间的适用协议中另有规定。甲骨文公司及其附属机构不对因您访问或使用第三方内容、产品或服务而产生的任何损失、费用或损害负责，除非您与甲骨文公司之间的适用协议中有规定。

本文档不是根据GPL许可证分发的。使用本文档须遵守以下条款。

你可以为你自己的个人使用而创建本文档的打印副本。只要不以任何方式改变或编辑实际内容，就允许转换为其他格式。您不得以任何形式或在任何媒体上发表或传播本文档，除非您以类似于甲骨文公司传播文档的方式（即在网站上与软件一起以电子方式下载）或在CD-ROM或类似的媒介上传播，但前提是该文档是与软件一起在同一媒介上传播的。任何其他用途，如任何印刷品的传播或在其他出版物中使用本文档的全部或部分内容，都需要事先获得甲骨文公司授权代表的书面同意。甲骨文公司和/或其附属机构保留对本文档的任何和所有权利，而不是上面明确授予的。

**文件的可访问性**

有关甲骨文公司对可访问性的承诺的信息，请访问甲骨文公司可访问性计划网站：https://www.oracle.com/corporate/accessibility/ 。

**获得甲骨文支持**

购买了支持的甲骨文客户可以通过我的甲骨文支持获得电子支持。有关信息，请访问https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab

目录

[第1章 概论 14](#_Toc70523050)

[1.1关于本手册 16](#_Toc70523051)

[1.2 MySQL数据库管理系统概览 21](#_Toc70523052)

[1.2.1 什么是MySQL 21](#_Toc70523053)

[1.2.2 MySQL的主要特性 25](#_Toc70523054)

[1.2.3 MySQL的历史 31](#_Toc70523055)

[1.3 MySQL 8.0中的新内容 32](#_Toc70523056)

[1.4 MySQL 8.0中增加、废弃或删除的服务器和状态变量及选项 113](#_Toc70523057)

[1.5 MySQL的信息来源 155](#_Toc70523058)

[1.6 如何报告缺陷或问题 157](#_Toc70523059)

[1.7 遵守MySQL标准 167](#_Toc70523060)

[1.7.1 MySQL对标准SQL的扩展 169](#_Toc70523061)

[1.7.2 MySQL与标准SQL的区别 175](#_Toc70523062)

[1.7.3 MySQL如何处理约束条件 180](#_Toc70523063)

[1.8 贡献者 184](#_Toc70523064)

[1.8.1 MySQL的贡献者 184](#_Toc70523065)

[1.8.2 文档作者和翻译者 189](#_Toc70523066)

[1.8.3 支持MySQL的软件包 191](#_Toc70523067)

[1.8.4 用于创建MySQL的工具 192](#_Toc70523068)

[1.8.5 MySQL的支持者 193](#_Toc70523069)

[第2章 安装或升级MySQL 194](#_Toc70523070)

[2.1 一般安装指南 197](#_Toc70523071)

[2.1.1 支持的平台 197](#_Toc70523072)

[2.1.2 安装哪个MySQL版本和发行版 197](#_Toc70523073)

[2.1.3 如何获得MySQL 200](#_Toc70523074)

[2.1.4 使用MD5校验和或GnuPG验证软件包的完整性 200](#_Toc70523075)

[2.1.5 安装布局 221](#_Toc70523076)

[2.1.6 特定于编译器的构建特征 221](#_Toc70523077)

[2.2 使用通用二进制文件在Unix/Linux上安装MySQL 222](#_Toc70523078)

[2.3 在Microsoft Windows上安装MySQL 227](#_Toc70523079)

[2.3.1 Microsoft Windows上的MySQL安装布局 232](#_Toc70523080)

[2.3.2 选择一个安装包 233](#_Toc70523081)

[2.3.3 用于Windows的MySQL安装程序 236](#_Toc70523082)

[2.3.4 使用noinstall ZIP档案在Microsoft Windows上安装MySQL 278](#_Toc70523083)

[2.3.5 Windows系统MySQL服务器安装的故障排除 294](#_Toc70523084)

[2.3.6 Windows安装后程序 297](#_Toc70523085)

[2.3.7 Windows平台的限制 301](#_Toc70523086)

[2.4 在macOS上安装MySQL 303](#_Toc70523087)

[2.4.1 在macOS上安装MySQL的一般注意事项 304](#_Toc70523088)

[2.4.2 使用本地软件包在macOS上安装MySQL 306](#_Toc70523089)

[2.4.3 安装和使用MySQL启动守护程序 312](#_Toc70523090)

[2.4.4 安装和使用MySQL首选项面板 318](#_Toc70523091)

[2.5 在Linux上安装MySQL 322](#_Toc70523092)

[2.5.1 使用MySQL Yum资源库在Linux上安装MySQL 324](#_Toc70523093)

[2.5.2 使用MySQL APT资源库在Linux上安装MySQL 331](#_Toc70523094)

[2.5.3 使用MySQL SLES资源库在Linux上安装MySQL 332](#_Toc70523095)

[2.5.4 使用Oracle的RPM包在Linux上安装MySQL 332](#_Toc70523096)

[2.5.5 使用Oracle的Debian软件包在Linux上安装MySQL 340](#_Toc70523097)

[2.5.6 用Docker在Linux上部署MySQL 342](#_Toc70523098)

[2.6 使用Unbreakable Linux Network（ULN）安装MySQL 380](#_Toc70523099)

[2.7 在Solaris上安装MySQL 382](#_Toc70523100)

[2.7.1 使用Solaris PKG在Solaris上安装MySQL 384](#_Toc70523101)

[2.8 在FreeBSD上安装MySQL 386](#_Toc70523102)

[2.9 从源代码安装MySQL 387](#_Toc70523103)

[2.9.1 源码安装方法 388](#_Toc70523104)

[2.9.2 源码安装的先决条件 389](#_Toc70523105)

[2.9.3 源码安装的MySQL布局 392](#_Toc70523106)

[2.9.4 使用标准源发布版安装MySQL 392](#_Toc70523107)

[2.9.5 使用开发源码树安装MySQL 399](#_Toc70523108)

[2.9.6 配置SSL库支持 402](#_Toc70523109)

[2.9.7 MySQL源配置选项 404](#_Toc70523110)

[2.10 安装后的设置和测试 457](#_Toc70523111)

[2.10.1 初始化数据目录 458](#_Toc70523112)

[2.10.2 启动服务器 468](#_Toc70523113)

[2.11 升级MySQL 483](#_Toc70523114)

[2.11.7 使用MySQL Yum仓库升级MySQL 543](#_Toc70523115)

[2.11.12 升级故障排除 550](#_Toc70523116)

[2.12 降级MySQL 557](#_Toc70523117)

[2.13 Perl安装注意事项 557](#_Toc70523118)

[2.13.1 在Unix上安装Perl 557](#_Toc70523119)

[2.13.2 在Windows上安装ActiveState Perl 559](#_Toc70523120)

[2.13.3 使用Perl DBI/DBD接口的问题 560](#_Toc70523121)

[第3章 指南 562](#_Toc70523122)

[第4章 MySQL程序 562](#_Toc70523123)

[第5章 MySQL服务管理 563](#_Toc70523124)

[第6章 安全 563](#_Toc70523125)

[第7章 备份与恢复 563](#_Toc70523126)

[第8章 优化 563](#_Toc70523127)

[第9章 语言结构 563](#_Toc70523128)

[第10章 字符集、对齐方式、统一编码 563](#_Toc70523129)

[第11章 数据类型 563](#_Toc70523130)

[第12章 函数和operator 563](#_Toc70523131)

[第13章 SQL表达式 563](#_Toc70523132)

[第14章 MySQL数据目录 564](#_Toc70523133)

[14.1 数据字典模式 565](#_Toc70523134)

[14.2 移除基于文件的元数据存储 567](#_Toc70523135)

[14.3 词典数据的事务性存储 568](#_Toc70523136)

[14.4 字典对象缓存 569](#_Toc70523137)

[14.5 INFORMATION\_SCHEMA和数据字典集成 570](#_Toc70523138)

[14.6 串行化字典信息（SDI 573](#_Toc70523139)

[14.7 数据字典使用方面的差异 574](#_Toc70523140)

[14.8 数据字典的限制 578](#_Toc70523141)

[第15章 InnoDB存储引擎 579](#_Toc70523142)

[第16章 其他存储引擎 579](#_Toc70523143)

[第17章 复制 579](#_Toc70523144)

[第18章 分组复制 579](#_Toc70523145)

[第19章 MySQL SHELL 580](#_Toc70523146)

[第20章 MySQL文档存储 582](#_Toc70523147)

[第21章 InnoDB 集群 582](#_Toc70523148)

[第22章 InnoDB 副本集 586](#_Toc70523149)

[第23章 MySQL NDB 集群 8.0 589](#_Toc70523150)

[第24章 分区 589](#_Toc70523151)

[第25章 存储对象 589](#_Toc70523152)

[第26章 INFORMATION 表 589](#_Toc70523153)

[第27章 MySQL性能schema 589](#_Toc70523154)

[第28章 MySQL sys schema 589](#_Toc70523155)

[第29章 连接器与API 590](#_Toc70523156)

[29.1 MySQL Connector/C++ 596](#_Toc70523157)

[29.2 MySQL Connector/J 596](#_Toc70523158)

[29.3 MySQL Connector/NET 596](#_Toc70523159)

[29.4 MySQL Connector/ODBC 596](#_Toc70523160)

[29.5 MySQL Connector/Python 597](#_Toc70523161)

[29.6 MySQL Connector/Node.js 597](#_Toc70523162)

[29.7 MySQL C API 597](#_Toc70523163)

[29.8 MySQL的PHP API 597](#_Toc70523164)

[29.9 MySQL的Perl API 597](#_Toc70523165)

[29.10 MySQL Python API 599](#_Toc70523166)

[29.11 MySQL Ruby APIs 599](#_Toc70523167)

[29.11.1 MySQL/Ruby API 600](#_Toc70523168)

[29.11.2 Ruby/MySQL API 600](#_Toc70523169)

[29.12 MySQL Tcl API 600](#_Toc70523170)

[29.13 MySQL Eiffel Wrapper 600](#_Toc70523171)

[第30章 MySQL企业版 601](#_Toc70523172)

[30.1 MySQL企业监控器概述 601](#_Toc70523173)

[30.2 MySQL企业备份概述 604](#_Toc70523174)

[30.3 MySQL企业版安全概述 605](#_Toc70523175)

[30.4 MySQL企业版加密概述 606](#_Toc70523176)

[30.5 MySQL企业审计概述 607](#_Toc70523177)

[30.6 MySQL企业防火墙概述 607](#_Toc70523178)

[30.7 MySQL企业线程池概述 608](#_Toc70523179)

[30.8 MySQL企业版数据屏蔽和取消识别概述 608](#_Toc70523180)

[第31章 MySQL WORKBENCH 610](#_Toc70523181)

[第32章 MySQL OCI 市场 612](#_Toc70523182)

[32.1 在Oracle云基础设施上部署MySQL EE的先决条件 612](#_Toc70523183)

[32.2 在Oracle云基础设施上部署MySQL EE 613](#_Toc70523184)

[32.3 配置网络访问 616](#_Toc70523185)

[32.4 连接 616](#_Toc70523186)

[32.5 维护 617](#_Toc70523187)

[附录A 619](#_Toc70523188)

[附录B 619](#_Toc70523189)

[附录C 619](#_Toc70523190)

[MySQL词汇表 619](#_Toc70523191)

# 指南

# MySQL数据目录

MySQL服务器集成了一个事务型数据字典，用于存储有关数据库对象的信息。在以前的MySQL版本中，字典数据存储在元数据文件、非事务性表和特定存储引擎的数据字典中。

本章描述了数据字典的主要特征、好处、使用差异和限制。关于数据字典功能的其他含义，请参阅MySQL 8.0发布说明中的 "数据字典说明 "部分。

MySQL数据字典的好处包括

* 集中式数据字典模式的简单性，统一存储字典数据。见第14.1节，"数据字典模式"。
* 消除了基于文件的元数据存储。参见第 14.2 节，"去除基于文件的元数据存储"。
* 对字典数据进行事务性的、防崩溃的存储。参见第 14.3 节，"字典数据的事务性存储"。
* 统一的、集中的字典对象的缓存。参见第 14.4 节，"字典对象的缓存"。
* 对一些INFORMATION\_SCHEMA表进行了更简单和改进的实现。参见第 14.5 节 "INFORMATION\_SCHEMA 和数据字典集成"。
* 原子型DDL。见第13.1.1节，"原子数据定义语句支持"。

重要的是

*与没有数据字典的服务器相比，启用了数据字典的服务器需要一些一般的操作差异；见第14.7节，"数据字典使用差异"。此外，对于升级到MySQL 8.0，升级过程与以前的MySQL版本有些不同，需要你通过检查特定的先决条件来验证你的安装是否准备好升级。欲了解更多信息，请参阅第2.11节 "升级MySQL"，特别是第2.11.5节 "为升级准备你的安装"。*

## 14.1 数据字典模式

数据字典表是受保护的，只能在MySQL的调试构建中访问。然而，MySQL支持通过INFORMATION\_SCHEMA表和SHOW语句访问存储在数据字典表中的数据。关于构成数据字典的表的概述，见数据字典表。

MySQL系统表在MySQL 8.0中仍然存在，可以通过在mysql系统数据库上发出SHOW TABLES语句来查看。一般来说，MySQL数据字典表和系统表之间的区别是，数据字典表包含执行SQL查询所需的元数据，而系统表包含辅助数据，如时区和帮助信息。MySQL系统表和数据字典表在升级方式上也有所不同。MySQL服务器负责管理数据字典的升级。SQL服务器。见数据字典如何升级。升级MySQL系统表需要运行完整的MySQL升级程序。参见第2.11.3节，"MySQL升级过程升级的内容"。

**数据字典是如何升级的**

新版本的MySQL可能包括对数据字典表定义的更改。这种变化存在于新安装的MySQL版本中，但当执行MySQL二进制文件的就地升级时，当使用新的二进制文件重新启动MySQL服务器时，就会应用这些变化。在启动时，服务器的数据字典版本与存储在数据字典中的版本信息进行比较，以确定是否应该升级数据字典表。如果升级是必要的并且被支持，服务器会用更新的定义创建数据字典表，把持久化的元数据复制到新的表中，用新的表原子化地替换旧的表，并重新初始化数据字典。如果不需要升级，则继续启动而不更新数据字典表。

数据字典表的升级是一个原子操作，这意味着所有的数据字典表都会根据需要进行升级，否则操作会失败。如果升级操作失败，服务器启动时就会出现错误。在这种情况下，可以使用旧的服务器二进制文件和旧的数据目录来启动服务器。当再次使用新的服务器二进制文件来启动服务器时，就会重新尝试进行数据字典的升级。

一般来说，在数据字典表成功升级后，不可能用旧的服务器二进制文件重新启动服务器。因此，在数据字典表被升级后，不支持将MySQL服务器二进制文件降级到以前的MySQL版本。

mysqld --no-dd-upgrade选项可以用来阻止启动时自动升级数据字典表。当--no-dd-upgrade被指定时，如果服务器发现服务器的数据字典版本与存储在数据字典中的版本不同，则启动失败，出现一个错误，说明禁止数据字典升级。

**使用MySQL的调试版本查看数据字典表**

数据字典表默认是受保护的，但可以通过编译支持调试的MySQL（使用-DWITH\_DEBUG=1 CMake选项）并指定+d,skip\_dd\_table\_access\_check调试选项和修改器来访问。关于编译调试构建的信息，见第5.9.1.1节，"编译调试的MySQL"。

警告

*不建议直接修改或写入数据字典表，这可能使你的MySQL实例无法运行。*

在用调试支持编译MySQL后，使用这个SET语句使数据字典表对mysql客户会话可见。

mysql> **SET SESSION debug='+d,skip\_dd\_table\_access\_check';**

使用这个查询来检索数据字典表的列表。

mysql> **SELECT name, schema\_id, hidden, type FROM mysql.tables where schema\_id=1 AND hidden='System';**

使用SHOW CREATE TABLE来查看数据字典表的定义。例如

mysql> **SHOW CREATE TABLE mysql.catalogs\G**

## 14.2 移除基于文件的元数据存储

在以前的MySQL版本中，字典数据被部分地存储在元数据文件中。基于文件的元数据存储的问题包括昂贵的文件扫描、易受文件系统相关错误的影响、处理复制和崩溃恢复故障状态的复杂代码，以及缺乏可扩展性，难以为新功能和关系对象添加元数据。

下面列出的元数据文件已从MySQL中删除。除非另有说明，以前存储在元数据文件中的数据现在存储在数据字典表中。

* .frm文件。表元数据文件。随着.frm文件的删除。
  + frm文件结构所规定的64KB的表定义大小限制被取消。
  + INFORMATION\_SCHEMA.TABLES VERSION列报告一个硬编码值10，这是MySQL 5.7中使用的最后一个.frm文件版本。
* .par文件。分区定义文件。InnoDB在MySQL 5.7中停止使用分区定义文件，因为引入了对InnoDB表的本地分区支持。
* .TRN文件。触发器命名空间文件。
* .TRG文件。触发器参数文件。
* .isl文件。InnoDB符号链接文件，包含了在数据目录之外创建的每个表的文件空间的位置。
* db.opt文件。数据库配置文件。这些文件，每个数据库目录一个，包含数据库默认的字符集属性。
* ddl\_log.log文件。该文件包含由数据定义语句产生的元数据操作记录，例如DROP TABLE和ALTER TABLE。

## 14.3 词典数据的事务性存储

数据字典模式在事务性（InnoDB）表中存储字典数据。数据字典表与非数据字典系统表一起位于mysql数据库中。

数据字典表在一个名为mysql.ibd的单一InnoDB表空间中创建，该表空间位于MySQL数据目录中。mysql.ibd表空间文件必须驻留在MySQL数据目录中，其名称不能被修改或被其他表空间使用。

字典数据受到与保护存储在InnoDB表中的用户数据相同的提交、回滚和崩溃恢复能力的保护。

## 14.4 字典对象缓存

字典对象缓存是一个共享的全局缓存，它将以前访问过的数据字典对象存储在内存中，以实现对象的重用，并使磁盘I/O最小化。与MySQL使用的其他缓存机制类似，字典对象缓存使用基于LRU的驱逐策略，从内存中驱逐最近使用的最小对象。

字典对象缓存包括存储不同对象类型的缓存分区。一些缓存分区的大小限制是可配置的，而另一些是硬编码的。

* 表空间定义高速缓存分区。存储表空间定义对象。tablespace\_definition\_cache选项为可以存储在字典对象缓存中的表空间定义对象的数量设置了一个限制。默认值是256。
* 模式定义缓存分区。存储模式定义对象。schema\_definition\_cache选项为可以存储在字典对象缓存中的模式定义对象的数量设置了一个限制。默认值是256。
* 表定义缓存分区。存储表定义对象。对象的限制被设置为max\_connections的值，它的默认值是151。

表定义缓存分区与使用table\_definition\_cache配置选项配置的表定义缓存平行存在。两个缓存都存储表定义，但为MySQL服务器的不同部分服务。一个缓存中的对象不依赖于另一个缓存中对象的存在。

* 存储程序定义缓存分区。存储存储的程序定义对象。stored\_program\_definition\_cache选项为可以存储在字典对象缓存中的存储程序定义对象的数量设定了一个限制。默认值是256。

存储程序定义缓存分区与存储过程和存储函数缓存并行存在，后者是通过store\_program\_cache选项配置的。

stored\_program\_cache选项为每个连接的存储过程或函数的缓存数量设置了一个软上限，并且在每次连接执行存储过程或函数时检查这个上限。另一方面，存储程序定义缓存分区是一个共享的缓存，用于存储其他用途的存储程序定义对象。存储程序定义缓存分区中对象的存在与存储过程缓存或存储函数缓存中对象的存在没有关系，反之亦然。

字符集定义缓存分区。存储字符集定义对象，硬编码的对象限制为256。

* 整理定义缓存分区。存储整理定义对象，硬编码的对象限制为256。

关于字典对象缓存配置选项的有效值，请参阅第5.1.8节，"服务器系统变量"。

## 14.5 INFORMATION\_SCHEMA和数据字典集成

随着数据字典的引入，以下INFORMATION\_SCHEMA表被实现为数据字典表的视图。

* [**CHARACTER\_SETS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table)
* [**CHECK\_CONSTRAINTS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-check-constraints-table)
* [**COLLATIONS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table)
* [**COLLATION\_CHARACTER\_SET\_APPLICABILITY**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collation-character-set-applicability-table)
* [**COLUMNS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-columns-table)
* [**COLUMN\_STATISTICS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-column-statistics-table)
* [**EVENTS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table)
* [**FILES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table)
* [**INNODB\_COLUMNS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-columns-table)
* [**INNODB\_DATAFILES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-datafiles-table)
* [**INNODB\_FIELDS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-fields-table)
* [**INNODB\_FOREIGN**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table)
* [**INNODB\_FOREIGN\_COLS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table)
* [**INNODB\_INDEXES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table)
* [**INNODB\_TABLES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table)
* [**INNODB\_TABLESPACES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table)
* [**INNODB\_TABLESPACES\_BRIEF**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-brief-table)
* [**INNODB\_TABLESTATS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table)
* [**KEY\_COLUMN\_USAGE**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-key-column-usage-table)
* **KEYWORDS**
* [**PARAMETERS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-parameters-table)
* [**PARTITIONS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-partitions-table)
* [**REFERENTIAL\_CONSTRAINTS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-referential-constraints-table)
* [**RESOURCE\_GROUPS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-resource-groups-table)
* [**ROUTINES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-routines-table)
* [**SCHEMATA**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table)
* [**STATISTICS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table)
* [**ST\_GEOMETRY\_COLUMNS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-geometry-columns-table)
* [**ST\_SPATIAL\_REFERENCE\_SYSTEMS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-spatial-reference-systems-table)
* [**TABLES**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table)
* [**TABLE\_CONSTRAINTS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-table-constraints-table)
* [**TRIGGERS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-triggers-table)
* [**VIEWS**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-views-table)
* [**VIEW\_ROUTINE\_USAGE**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-view-routine-usage-table)
* [**VIEW\_TABLE\_USAGE**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-view-table-usage-table)

对这些表的查询现在更有效率了，因为他们从数据字典表获得信息，而不是通过其他更慢的方式。特别是对于每个INFORMATION\_SCHEMA表，它是数据字典表上的一个视图。

* 服务器不再必须为INFORMATION\_SCHEMA表的每次查询创建一个临时表。
* 当底层数据字典表存储了以前通过目录扫描（例如，列举数据库名称或数据库内的表名称）或文件打开操作（例如，从.frm文件中读取信息）获得的值时，INFORMATION\_SCHEMA查询这些值现在使用表查找。(另外，即使对于非视图的INFORMATION\_SCHEMA表，数据库和表名等值也是通过数据字典的查找来获取的，不需要目录或文件扫描。)
* 在底层数据字典表上的索引允许优化器构建有效的查询执行计划，这在以前的实现中是不存在的，它是通过每个查询使用一个临时表来处理 INFORMATION\_SCHEMA 表。

前面的改进也适用于显示对应于INFORMATION\_SCHEMA表的信息的SHOW语句，这些表是数据字典表的视图。例如，SHOW DATABASES显示与SCHEMATA表相同的信息。

除了在数据字典表上引入视图之外，在STATISTICS和TABLES表中包含的表的统计数据现在被缓存起来，以提高INFORMATION\_SCHEMA的查询性能。information\_schema\_stats\_expiry系统变量定义了缓存的表统计信息过期前的时间段。默认是86400秒（24小时）。如果没有缓存的统计数据或者统计数据已经过期，那么在查询表的统计列时，会从存储引擎中检索统计数据。要在任何时候更新某个表的缓存值，请使用ANALYZE TABLE

information\_schema\_stats\_expiry可以被设置为0，让INFORMATION\_SCHEMA查询直接从存储引擎检索最新的统计数据，这不如检索缓存的统计数据快。

更多信息请参见第8.2.3节，"优化INFORMATION\_SCHEMA查询"。

在MySQL 8.0中，INFORMATION\_SCHEMA表与数据字典紧密联系在一起，导致了一些使用上的差异。见第14.7节，"数据字典使用差异"。

## 14.6 串行化字典信息（SDI

除了在数据字典中存储关于数据库对象的元数据外，MySQL还以序列化的形式存储它。这种数据被称为序列化字典信息（SDI）。InnoDB在其表空间文件中存储SDI数据。NDBCLUSTER将SDI数据存储在NDB字典中。其他存储引擎将SDI数据存储在.sdi文件中，这些文件是在表的数据库目录下为一个给定的表创建的。SDI数据以一种紧凑的JSON格式生成。

串行化的字典信息（SDI）存在于所有InnoDB表空间文件中，除了临时表空间和undo tablespaces文件。InnoDB表空间文件中的SDI记录只描述表空间中包含的表和表空间对象。

SDI数据通过对表的DDL操作或CHECK TABLE FOR UPGRADE进行更新。当MySQL服务器升级到一个新版本或版本时，SDI数据不会被更新。

SDI数据的存在提供了元数据的冗余性。例如，如果数据字典变得不可用，可以使用ibd2sdi工具直接从InnoDB表空间文件中提取对象元数据。

对于InnoDB，一个SDI记录需要一个索引页，默认大小为16KB。然而，SDI数据被压缩以减少存储空间的占用。

对于由多个表空间组成的分区InnoDB表，SDI数据被存储在第一个分区的表空间文件中。

MySQL服务器使用一个内部API，在DDL操作期间访问该API以创建和维护SDI记录。

IMPORT TABLE语句根据.sdi文件中包含的信息导入MyISAM表。更多信息，请参见第13.2.5节，"IMPORT TABLE语句"。

## 14.7 数据字典使用方面的差异

与没有数据字典的服务器相比，使用启用了数据字典的MySQL服务器会带来一些操作上的差异。

* 以前，启用innodb\_read\_only系统变量可以阻止只为InnoDB存储引擎创建和删除表。从MySQL 8.0开始，启用innodb\_read\_only会阻止所有存储引擎的这些操作。任何存储引擎的表创建和删除操作都会修改mysql系统数据库中的数据字典表，但这些表使用InnoDB存储引擎，在启用innodb\_read\_only时不能被修改。这个原则也适用于其他需要修改数据字典表的表操作。例子。
  + ANALYZE TABLE失败，因为它更新了存储在数据字典中的表的统计数据。
  + ALTER TABLE tbl\_name ENGINE=engine\_name失败，因为它更新了存储引擎的名称，而这个名称是存储在数据字典中的。

注意

*启用innodb\_read\_only对mysql系统数据库中的非数据字典表也有重要影响。详情请参见第15.14节 "InnoDB启动选项和系统变量 "中对innodb\_read\_only的描述。*

* 以前，mysql系统数据库中的表对于DML和DDL语句是可见的。从MySQL 8.0开始，数据字典表是不可见的，不能被直接修改或查询。然而，在大多数情况下，有相应的INFORMATION\_SCHEMA表可以被查询。这使得底层的数据字典表可以随着服务器开发的进行而改变，同时保持一个稳定的INFORMATION\_SCHEMA接口供应用程序使用。
* 在MySQL 8.0中，INFORMATION\_SCHEMA表与数据字典紧密相连，导致了一些使用上的差异。
  + 以前，INFORMATION\_SCHEMA查询STATISTICS和TABLES表中的表的统计数据，直接从存储引擎检索统计数据。从MySQL 8.0开始，默认使用缓存的表统计数据。information\_schema\_stats\_expiry系统变量定义了缓存表统计数据过期前的时间段。默认是86400秒（24小时）。要想在任何时候更新某个表的缓存值，请使用ANALYZE TABLE）。如果没有缓存的统计信息或者统计信息已经过期，在查询表的统计列时，会从存储引擎中检索统计信息。要想总是直接从存储引擎中获取最新的统计数据，请将information\_schema\_stats\_expiry设置为0。 更多信息请参见8.2.3节 "优化INFORMATION\_SCHEMA查询"。
  + 几个INFORMATION\_SCHEMA表是数据字典表的视图，这使得优化器可以在这些底层表上使用索引。因此，根据优化器的选择，INFORMATION\_SCHEMA查询结果的行顺序可能与之前的结果不同。如果一个查询结果必须有特定的行排序特征，请包括一个ORDER BY子句。
  + 对INFORMATION\_SCHEMA表的查询可能会返回与早期MySQL系列不同的字母大写的列名。应用程序应以不区分大小写的方式测试结果集列名。如果这不可行，一个变通方法是在选择列表中使用列别名，以要求的字母大小写返回列名。比如说。

SELECT TABLE\_SCHEMA AS table\_schema, TABLE\_NAME AS table\_name

FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'users';

* mysqldump和mysqlpump不再转储INFORMATION\_SCHEMA数据库，即使在命令行中明确命名。
* CREATE TABLE dst\_tbl LIKE src\_tbl要求src\_tbl是一个基表，如果它是一个INFORMATION\_SCHEMA表，是数据字典表的视图，则失败。
* 以前，从INFORMATION\_SCHEMA表中选择的列的结果集标题使用查询中指定的大写字母。这个查询产生的结果集的标题是table\_name。

SELECT table\_name FROM INFORMATION\_SCHEMA.TABLES;

从MySQL 8.0开始，这些头是大写的；前面的查询产生一个头为TABLE\_NAME的结果集。如果有必要，可以用一个列的别名来实现不同的字母大小写。例如。

SELECT table\_name AS 'table\_name' FROM INFORMATION\_SCHEMA.TABLES;

* 数据目录会影响mysqldump和mysqlpump从mysql系统数据库转储信息的方式。
* 以前，可以转储mysql系统数据库中的所有表。从MySQL 8.0开始，mysqldump和mysqlpump只转储该数据库中的非数据字典表。
  + 以前，可以转储mysql系统数据库中的所有表。从MySQL 8.0开始，mysqldump和mysqlpump只转储该数据库中的非数据字典表。
  + 以前，当使用--all-databases选项时，不需要--routines和--events选项来包括存储例程和事件。转储包括mysql系统数据库，因此也包括含有存储例程和事件定义的proc和事件表。从MySQL 8.0开始，事件表和proc表不被使用。相应对象的定义被存储在数据字典表中，但这些表没有被转储。要在使用--all-databases进行的转储中包括存储的例程和事件，请明确使用--routines和--events选项。
  + 以前， --routines选项需要proc表的SELECT权限。从MySQL 8.0开始，该表不被使用；--routines需要全局SELECT权限来代替。
  + 以前，可以通过转储proc和event表来转储存储的例程和事件定义，以及它们的创建和修改时间戳。从MySQL 8.0开始，这些表不被使用，所以不可能转储时间戳。
* 以前，创建一个包含非法字符的存储例程会产生一个警告。从MySQL 8.0开始，这就是一个错误。

## 14.8 数据字典的限制

本节介绍了用MySQL数据字典引入的临时限制。

* 不支持在数据目录下手动创建数据库目录（例如，用mkdir）。手动创建的数据库目录不被MySQL服务器所识别。
* 由于向存储、Undo logs和redo log而不是.frm文件写入，DDL操作需要更长的时间。

# InnoDB存储引擎

## 15.1 Introduction to InnoDB

[15.1.1 Benefits of Using InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-benefits)

[15.1.2 Best Practices for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-best-practices)

[15.1.3 Verifying that InnoDB is the Default Storage Engine](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-check-availability)

[15.1.4 Testing and Benchmarking with InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-benchmarking)

**InnoDB** is a general-purpose storage engine that balances high reliability and high performance. In MySQL 8.0, **InnoDB** is the default MySQL storage engine. Unless you have configured a different default storage engine, issuing a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement without an **ENGINE** clause creates an **InnoDB** table.

### Key Advantages of InnoDB

Its DML operations follow the ACID model, with transactions featuring commit, rollback, and crash-recovery capabilities to protect user data. See [Section 15.2, “InnoDB and the ACID Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#mysql-acid).

Row-level locking and Oracle-style consistent reads increase multi-user concurrency and performance. See [Section 15.7, “InnoDB Locking and Transaction Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-transaction-model).

**InnoDB** tables arrange your data on disk to optimize queries based on primary keys. Each **InnoDB** table has a primary key index called the clustered index that organizes the data to minimize I/O for primary key lookups. See [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types).

To maintain data integrity, **InnoDB** supports **FOREIGN KEY** constraints. With foreign keys, inserts, updates, and deletes are checked to ensure they do not result in inconsistencies across related tables. See [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

**Table 15.1 InnoDB Storage Engine Features**

| **Feature** | **Support** |
| --- | --- |
| ***B-tree indexes*** | Yes |
| ***Backup/point-in-time recovery*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Cluster database support*** | No |
| ***Clustered indexes*** | Yes |
| ***Compressed data*** | Yes |
| ***Data caches*** | Yes |
| ***Encrypted data*** | Yes (Implemented in the server via encryption functions; In MySQL 5.7 and later, data-at-rest encryption is supported.) |
| ***Foreign key support*** | Yes |
| ***Full-text search indexes*** | Yes (Support for FULLTEXT indexes is available in MySQL 5.6 and later.) |
| ***Geospatial data type support*** | Yes |
| ***Geospatial indexing support*** | Yes (Support for geospatial indexing is available in MySQL 5.7 and later.) |
| ***Hash indexes*** | No (InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.) |
| ***Index caches*** | Yes |
| ***Locking granularity*** | Row |
| ***MVCC*** | Yes |
| ***Replication support*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Storage limits*** | 64TB |
| ***T-tree indexes*** | No |
| ***Transactions*** | Yes |
| ***Update statistics for data dictionary*** | Yes |

To compare the features of **InnoDB** with other storage engines provided with MySQL, see the Storage Engine Features table in [Chapter 16, *Alternative Storage Engines*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html).

### InnoDB Enhancements and New Features

For information about **InnoDB** enhancements and new features, refer to:

The **InnoDB** enhancements list in [Section 1.3, “What Is New in MySQL 8.0”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#mysql-nutshell).

The [Release Notes](https://dev.mysql.com/doc/relnotes/mysql/8.0/en/).

### Additional InnoDB Information and Resources

For **InnoDB**-related terms and definitions, see the [MySQL Glossary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html).

For a forum dedicated to the **InnoDB** storage engine, see [MySQL Forums::InnoDB](http://forums.mysql.com/list.php?22).

**InnoDB** is published under the same GNU GPL License Version 2 (of June 1991) as MySQL. For more information on MySQL licensing, see <http://www.mysql.com/company/legal/licensing/>.

### 15.1.1 Benefits of Using InnoDB Tables

**InnoDB** tables have the following benefits:

If the server unexpectedly exits because of a hardware or software issue, regardless of what was happening in the database at the time, you don't need to do anything special after restarting the database. **InnoDB** crash recovery automatically finalizes changes that were committed before the time of the crash, and undoes changes that were in process but not committed, permitting you to restart and continue from where you left off. See [Section 15.18.2, “InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery).

The **InnoDB** storage engine maintains its own buffer pool that caches table and index data in main memory as data is accessed. Frequently used data is processed directly from memory. This cache applies to many types of information and speeds up processing. On dedicated database servers, up to 80% of physical memory is often assigned to the buffer pool. See [Section 15.5.1, “Buffer Pool”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool).

If you split up related data into different tables, you can set up foreign keys that enforce referential integrity. See [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

If data becomes corrupted on disk or in memory, a checksum mechanism alerts you to the bogus data before you use it. The [**innodb\_checksum\_algorithm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checksum_algorithm) variable defines the checksum algorithm used by **InnoDB**.

When you design a database with appropriate primary key columns for each table, operations involving those columns are automatically optimized. It is very fast to reference the primary key columns in [**WHERE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) clauses, [**ORDER BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) clauses, [**GROUP BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) clauses, and join operations. See [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types).

Inserts, updates, and deletes are optimized by an automatic mechanism called change buffering. **InnoDB** not only allows concurrent read and write access to the same table, it caches changed data to streamline disk I/O. See [Section 15.5.2, “Change Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer).

Performance benefits are not limited to large tables with long-running queries. When the same rows are accessed over and over from a table, the Adaptive Hash Index takes over to make these lookups even faster, as if they came out of a hash table. See [Section 15.5.3, “Adaptive Hash Index”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-adaptive-hash).

You can compress tables and associated indexes. See [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression).

You can encrypt your data. See [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

You can create and drop indexes and perform other DDL operations with much less impact on performance and availability. See [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations).

Truncating a file-per-table tablespace is very fast and can free up disk space for the operating system to reuse rather than only **InnoDB**. See [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces).

The storage layout for table data is more efficient for [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and long text fields, with the **DYNAMIC** row format. See [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

You can monitor the internal workings of the storage engine by querying **INFORMATION\_SCHEMA** tables. See [Section 15.15, “InnoDB INFORMATION\_SCHEMA Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema).

You can monitor the performance details of the storage engine by querying Performance Schema tables. See [Section 15.16, “InnoDB Integration with MySQL Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-schema).

You can mix **InnoDB** tables with tables from other MySQL storage engines, even within the same statement. For example, you can use a join operation to combine data from **InnoDB** and [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables in a single query.

**InnoDB** has been designed for CPU efficiency and maximum performance when processing large data volumes.

**InnoDB** tables can handle large quantities of data, even on operating systems where file size is limited to 2GB.

For **InnoDB**-specific tuning techniques you can apply to your MySQL server and application code, see [Section 8.5, “Optimizing for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb).

### 15.1.2 Best Practices for InnoDB Tables

This section describes best practices when using **InnoDB** tables.

Specify a primary key for every table using the most frequently queried column or columns, or an auto-increment value if there is no obvious primary key.

Use joins wherever data is pulled from multiple tables based on identical ID values from those tables. For fast join performance, define foreign keys on the join columns, and declare those columns with the same data type in each table. Adding foreign keys ensures that referenced columns are indexed, which can improve performance. Foreign keys also propagate deletes and updates to all affected tables, and prevent insertion of data in a child table if the corresponding IDs are not present in the parent table.

Turn off autocommit. Committing hundreds of times a second puts a cap on performance (limited by the write speed of your storage device).

Group sets of related DML operations into transactions by bracketing them with **START TRANSACTION** and **COMMIT** statements. While you don't want to commit too often, you also don't want to issue huge batches of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements that run for hours without committing.

Do not use [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statements. **InnoDB** can handle multiple sessions all reading and writing to the same table at once without sacrificing reliability or high performance. To get exclusive write access to a set of rows, use the [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads) syntax to lock just the rows you intend to update.

Enable the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable or use general tablespaces to put the data and indexes for tables into separate files instead of the system tablespace. The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable is enabled by default.

Evaluate whether your data and access patterns benefit from the **InnoDB** table or page compression features. You can compress **InnoDB** tables without sacrificing read/write capability.

Run the server with the [**--sql\_mode=NO\_ENGINE\_SUBSTITUTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) option to prevent tables from being created with storage engines that you do not want to use.

### 15.1.3 Verifying that InnoDB is the Default Storage Engine

Issue the [**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines) statement to view the available MySQL storage engines. Look for **DEFAULT** in the **SUPPORT** column.

mysql> SHOW ENGINES;

Alternatively, query the [**INFORMATION\_SCHEMA.ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-engines-table) table.

mysql> SELECT \* FROM INFORMATION\_SCHEMA.ENGINES;

### 15.1.4 Testing and Benchmarking with InnoDB

If **InnoDB** is not the default storage engine, you can determine if your database server and applications work correctly with **InnoDB** by restarting the server with [**--default-storage-engine=InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) defined on the command line or with [**default-storage-engine=innodb**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) defined in the **[mysqld]** section of the MySQL server option file.

Since changing the default storage engine only affects newly created tables, run your application installation and setup steps to confirm that everything installs properly, then exercise the application features to make sure the data loading, editing, and querying features work. If a table relies on a feature that is specific to another storage engine, you receive an error. In this case, add the **ENGINE=*other\_engine\_name*** clause to the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement to avoid the error.

If you did not make a deliberate decision about the storage engine, and you want to preview how certain tables work when created using **InnoDB**, issue the command [**ALTER TABLE table\_name ENGINE=InnoDB;**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) for each table. Alternatively, to run test queries and other statements without disturbing the original table, make a copy:

CREATE TABLE ... ENGINE=InnoDB AS SELECT \* FROM ***other\_engine\_table***;

To assess performance with a full application under a realistic workload, install the latest MySQL server and run benchmarks.

Test the full application lifecycle, from installation, through heavy usage, and server restart. Kill the server process while the database is busy to simulate a power failure, and verify that the data is recovered successfully when you restart the server.

Test any replication configurations, especially if you use different MySQL versions and options on the source server and replicas.

## 15.2 InnoDB and the ACID Model

The [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid) model is a set of database design principles that emphasize aspects of reliability that are important for business data and mission-critical applications. MySQL includes components such as the **InnoDB** storage engine that adhere closely to the ACID model so that data is not corrupted and results are not distorted by exceptional conditions such as software crashes and hardware malfunctions. When you rely on ACID-compliant features, you do not need to reinvent the wheel of consistency checking and crash recovery mechanisms. In cases where you have additional software safeguards, ultra-reliable hardware, or an application that can tolerate a small amount of data loss or inconsistency, you can adjust MySQL settings to trade some of the ACID reliability for greater performance or throughput.

The following sections discuss how MySQL features, in particular the **InnoDB** storage engine, interact with the categories of the ACID model:

***A***: atomicity.

***C***: consistency.

***I:***: isolation.

***D***: durability.

### Atomicity

The ***atomicity*** aspect of the ACID model mainly involves **InnoDB** [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction). Related MySQL features include:

The [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) setting.

The [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement.

The [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement.

### Consistency

The ***consistency*** aspect of the ACID model mainly involves internal **InnoDB** processing to protect data from crashes. Related MySQL features include:

The **InnoDB** doublewrite buffer. See [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**InnoDB** crash recovery. See [InnoDB Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-crash-recovery).

### Isolation

The ***isolation*** aspect of the ACID model mainly involves **InnoDB** [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction), in particular the [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) that applies to each transaction. Related MySQL features include:

The [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) setting.

Transaction isolation levels and the [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement. See [Section 15.7.2.1, “Transaction Isolation Levels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-isolation-levels).

The low-level details of **InnoDB** [locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking). Details can be viewed in the **INFORMATION\_SCHEMA** tables (see [Section 15.15.2, “InnoDB INFORMATION\_SCHEMA Transaction and Locking Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-transactions)) and Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables.

### Durability

The ***durability*** aspect of the ACID model involves MySQL software features interacting with your particular hardware configuration. Because of the many possibilities depending on the capabilities of your CPU, network, and storage devices, this aspect is the most complicated to provide concrete guidelines for. (And those guidelines might take the form of “buy new hardware”.) Related MySQL features include:

The **InnoDB** doublewrite buffer. See [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

The [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) variable.

The [**sync\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) variable.

The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable.

The write buffer in a storage device, such as a disk drive, SSD, or RAID array.

A battery-backed cache in a storage device.

The operating system used to run MySQL, in particular its support for the **fsync()** system call.

An uninterruptible power supply (UPS) protecting the electrical power to all computer servers and storage devices that run MySQL servers and store MySQL data.

Your backup strategy, such as frequency and types of backups, and backup retention periods.

For distributed or hosted data applications, the particular characteristics of the data centers where the hardware for the MySQL servers is located, and network connections between the data centers.

## 15.3 InnoDB Multi-Versioning

**InnoDB** is a multi-version storage engine. It keeps information about old versions of changed rows to support transactional features such as concurrency and rollback. This information is stored in undo tablespaces in a data structure called a rollback segment. See [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces). **InnoDB** uses the information in the rollback segment to perform the undo operations needed in a transaction rollback. It also uses the information to build earlier versions of a row for a consistent read. See [Section 15.7.2.3, “Consistent Nonlocking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-consistent-read).

Internally, **InnoDB** adds three fields to each row stored in the database:

A 6-byte **DB\_TRX\_ID** field indicates the transaction identifier for the last transaction that inserted or updated the row. Also, a deletion is treated internally as an update where a special bit in the row is set to mark it as deleted.

A 7-byte **DB\_ROLL\_PTR** field called the roll pointer. The roll pointer points to an undo log record written to the rollback segment. If the row was updated, the undo log record contains the information necessary to rebuild the content of the row before it was updated.

A 6-byte **DB\_ROW\_ID** field contains a row ID that increases monotonically as new rows are inserted. If **InnoDB** generates a clustered index automatically, the index contains row ID values. Otherwise, the **DB\_ROW\_ID** column does not appear in any index.

Undo logs in the rollback segment are divided into insert and update undo logs. Insert undo logs are needed only in transaction rollback and can be discarded as soon as the transaction commits. Update undo logs are used also in consistent reads, but they can be discarded only after there is no transaction present for which **InnoDB** has assigned a snapshot that in a consistent read could require the information in the update undo log to build an earlier version of a database row. For additional information about undo logs, see [Section 15.6.6, “Undo Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-logs).

It is recommend that you commit transactions regularly, including transactions that issue only consistent reads. Otherwise, **InnoDB** cannot discard data from the update undo logs, and the rollback segment may grow too big, filling up the undo tablespace in which it resides. For information about managing undo tablespaces, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

The physical size of an undo log record in the rollback segment is typically smaller than the corresponding inserted or updated row. You can use this information to calculate the space needed for your rollback segment.

In the **InnoDB** multi-versioning scheme, a row is not physically removed from the database immediately when you delete it with an SQL statement. **InnoDB** only physically removes the corresponding row and its index records when it discards the update undo log record written for the deletion. This removal operation is called a purge, and it is quite fast, usually taking the same order of time as the SQL statement that did the deletion.

If you insert and delete rows in smallish batches at about the same rate in the table, the purge thread can start to lag behind and the table can grow bigger and bigger because of all the “dead” rows, making everything disk-bound and very slow. In such cases, throttle new row operations, and allocate more resources to the purge thread by tuning the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) system variable. For more information, see [Section 15.8.9, “Purge Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration).

### Multi-Versioning and Secondary Indexes

**InnoDB** multiversion concurrency control (MVCC) treats secondary indexes differently than clustered indexes. Records in a clustered index are updated in-place, and their hidden system columns point undo log entries from which earlier versions of records can be reconstructed. Unlike clustered index records, secondary index records do not contain hidden system columns nor are they updated in-place.

When a secondary index column is updated, old secondary index records are delete-marked, new records are inserted, and delete-marked records are eventually purged. When a secondary index record is delete-marked or the secondary index page is updated by a newer transaction, **InnoDB** looks up the database record in the clustered index. In the clustered index, the record's **DB\_TRX\_ID** is checked, and the correct version of the record is retrieved from the undo log if the record was modified after the reading transaction was initiated.

If a secondary index record is marked for deletion or the secondary index page is updated by a newer transaction, the [covering index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_covering_index) technique is not used. Instead of returning values from the index structure, **InnoDB** looks up the record in the clustered index.

However, if the [index condition pushdown (ICP)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-condition-pushdown-optimization) optimization is enabled, and parts of the **WHERE** condition can be evaluated using only fields from the index, the MySQL server still pushes this part of the **WHERE** condition down to the storage engine where it is evaluated using the index. If no matching records are found, the clustered index lookup is avoided. If matching records are found, even among delete-marked records, **InnoDB** looks up the record in the clustered index.

## 15.4 InnoDB Architecture

The following diagram shows in-memory and on-disk structures that comprise the **InnoDB** storage engine architecture. For information about each structure, see [Section 15.5, “InnoDB In-Memory Structures”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-in-memory-structures), and [Section 15.6, “InnoDB On-Disk Structures”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-on-disk-structures).

**Figure 15.1 InnoDB Architecture**

## 15.5 InnoDB In-Memory Structures

[15.5.1 Buffer Pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool)

[15.5.2 Change Buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer)

[15.5.3 Adaptive Hash Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-adaptive-hash)

[15.5.4 Log Buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-redo-log-buffer)

This section describes **InnoDB** in-memory structures and related topics.

### 15.5.1 Buffer Pool

The buffer pool is an area in main memory where **InnoDB** caches table and index data as it is accessed. The buffer pool permits frequently used data to be accessed directly from memory, which speeds up processing. On dedicated servers, up to 80% of physical memory is often assigned to the buffer pool.

For efficiency of high-volume read operations, the buffer pool is divided into pages that can potentially hold multiple rows. For efficiency of cache management, the buffer pool is implemented as a linked list of pages; data that is rarely used is aged out of the cache using a variation of the least recently used (LRU) algorithm.

Knowing how to take advantage of the buffer pool to keep frequently accessed data in memory is an important aspect of MySQL tuning.

#### Buffer Pool LRU Algorithm

The buffer pool is managed as a list using a variation of the LRU algorithm. When room is needed to add a new page to the buffer pool, the least recently used page is evicted and a new page is added to the middle of the list. This midpoint insertion strategy treats the list as two sublists:

At the head, a sublist of new (“young”) pages that were accessed recently

At the tail, a sublist of old pages that were accessed less recently

**Figure 15.2 Buffer Pool List**

The algorithm keeps frequently used pages in the new sublist. The old sublist contains less frequently used pages; these pages are candidates for [eviction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction).

By default, the algorithm operates as follows:

3/8 of the buffer pool is devoted to the old sublist.

The midpoint of the list is the boundary where the tail of the new sublist meets the head of the old sublist.

When **InnoDB** reads a page into the buffer pool, it initially inserts it at the midpoint (the head of the old sublist). A page can be read because it is required for a user-initiated operation such as an SQL query, or as part of a [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) operation performed automatically by **InnoDB**.

Accessing a page in the old sublist makes it “young”, moving it to the head of the new sublist. If the page was read because it was required by a user-initiated operation, the first access occurs immediately and the page is made young. If the page was read due to a read-ahead operation, the first access does not occur immediately and might not occur at all before the page is evicted.

As the database operates, pages in the buffer pool that are not accessed “age” by moving toward the tail of the list. Pages in both the new and old sublists age as other pages are made new. Pages in the old sublist also age as pages are inserted at the midpoint. Eventually, a page that remains unused reaches the tail of the old sublist and is evicted.

By default, pages read by queries are immediately moved into the new sublist, meaning they stay in the buffer pool longer. A table scan, performed for a [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) operation or a **SELECT** statement with no **WHERE** clause, for example, can bring a large amount of data into the buffer pool and evict an equivalent amount of older data, even if the new data is never used again. Similarly, pages that are loaded by the read-ahead background thread and accessed only once are moved to the head of the new list. These situations can push frequently used pages to the old sublist where they become subject to eviction. For information about optimizing this behavior, see [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion), and [Section 15.8.3.4, “Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead).

**InnoDB** Standard Monitor output contains several fields in the **BUFFER POOL AND MEMORY** section regarding operation of the buffer pool LRU algorithm. For details, see [Monitoring the Buffer Pool Using the InnoDB Standard Monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-monitoring).

#### Buffer Pool Configuration

You can configure the various aspects of the buffer pool to improve performance.

Ideally, you set the size of the buffer pool to as large a value as practical, leaving enough memory for other processes on the server to run without excessive paging. The larger the buffer pool, the more **InnoDB** acts like an in-memory database, reading data from disk once and then accessing the data from memory during subsequent reads. See [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize).

On 64-bit systems with sufficient memory, you can split the buffer pool into multiple parts to minimize contention for memory structures among concurrent operations. For details, see [Section 15.8.3.2, “Configuring Multiple Buffer Pool Instances”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multiple-buffer-pools).

You can keep frequently accessed data in memory regardless of sudden spikes of activity from operations that would bring large amounts of infrequently accessed data into the buffer pool. For details, see [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion).

You can control how and when to perform read-ahead requests to prefetch pages into the buffer pool asynchronously in anticipation of impending need for them. For details, see [Section 15.8.3.4, “Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead).

You can control when background flushing occurs and whether or not the rate of flushing is dynamically adjusted based on workload. For details, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

You can configure how **InnoDB** preserves the current buffer pool state to avoid a lengthy warmup period after a server restart. For details, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

#### Monitoring the Buffer Pool Using the InnoDB Standard Monitor

**InnoDB** Standard Monitor output, which can be accessed using [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor), provides metrics regarding operation of the buffer pool. Buffer pool metrics are located in the **BUFFER POOL AND MEMORY** section of **InnoDB** Standard Monitor output:

----------------------

BUFFER POOL AND MEMORY

----------------------

Total large memory allocated 2198863872

Dictionary memory allocated 776332

Buffer pool size 131072

Free buffers 124908

Database pages 5720

Old database pages 2071

Modified db pages 910

Pending reads 0

Pending writes: LRU 0, flush list 0, single page 0

Pages made young 4, not young 0

0.10 youngs/s, 0.00 non-youngs/s

Pages read 197, created 5523, written 5060

0.00 reads/s, 190.89 creates/s, 244.94 writes/s

Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not

0 / 1000

Pages read ahead 0.00/s, evicted without access 0.00/s, Random read

ahead 0.00/s

LRU len: 5720, unzip\_LRU len: 0

I/O sum[0]:cur[0], unzip sum[0]:cur[0]

The following table describes buffer pool metrics reported by the **InnoDB** Standard Monitor.

Per second averages provided in **InnoDB** Standard Monitor output are based on the elapsed time since **InnoDB** Standard Monitor output was last printed.

**Table 15.2 InnoDB Buffer Pool Metrics**

| **Name** | **Description** |
| --- | --- |
| Total memory allocated | The total memory allocated for the buffer pool in bytes. |
| Dictionary memory allocated | The total memory allocated for the **InnoDB** data dictionary in bytes. |
| Buffer pool size | The total size in pages allocated to the buffer pool. |
| Free buffers | The total size in pages of the buffer pool free list. |
| Database pages | The total size in pages of the buffer pool LRU list. |
| Old database pages | The total size in pages of the buffer pool old LRU sublist. |
| Modified db pages | The current number of pages modified in the buffer pool. |
| Pending reads | The number of buffer pool pages waiting to be read into the buffer pool. |
| Pending writes LRU | The number of old dirty pages within the buffer pool to be written from the bottom of the LRU list. |
| Pending writes flush list | The number of buffer pool pages to be flushed during checkpointing. |
| Pending writes single page | The number of pending independent page writes within the buffer pool. |
| Pages made young | The total number of pages made young in the buffer pool LRU list (moved to the head of sublist of “new” pages). |
| Pages made not young | The total number of pages not made young in the buffer pool LRU list (pages that have remained in the “old” sublist without being made young). |
| youngs/s | The per second average of accesses to old pages in the buffer pool LRU list that have resulted in making pages young. See the notes that follow this table for more information. |
| non-youngs/s | The per second average of accesses to old pages in the buffer pool LRU list that have resulted in not making pages young. See the notes that follow this table for more information. |
| Pages read | The total number of pages read from the buffer pool. |
| Pages created | The total number of pages created within the buffer pool. |
| Pages written | The total number of pages written from the buffer pool. |
| reads/s | The per second average number of buffer pool page reads per second. |
| creates/s | The average number of buffer pool pages created per second. |
| writes/s | The average number of buffer pool page writes per second. |
| Buffer pool hit rate | The buffer pool page hit rate for pages read from the buffer pool vs from disk storage. |
| young-making rate | The average hit rate at which page accesses have resulted in making pages young. See the notes that follow this table for more information. |
| not (young-making rate) | The average hit rate at which page accesses have not resulted in making pages young. See the notes that follow this table for more information. |
| Pages read ahead | The per second average of read ahead operations. |
| Pages evicted without access | The per second average of the pages evicted without being accessed from the buffer pool. |
| Random read ahead | The per second average of random read ahead operations. |
| LRU len | The total size in pages of the buffer pool LRU list. |
| unzip\_LRU len | The length (in pages) of the buffer pool unzip\_LRU list. |
| I/O sum | The total number of buffer pool LRU list pages accessed. |
| I/O cur | The total number of buffer pool LRU list pages accessed in the current interval. |
| I/O unzip sum | The total number of buffer pool unzip\_LRU list pages decompressed. |
| I/O unzip cur | The total number of buffer pool unzip\_LRU list pages decompressed in the current interval. |

***Notes***:

The **youngs/s** metric is applicable only to old pages. It is based on the number of page accesses. There can be multiple accesses for a given page, all of which are counted. If you see very low **youngs/s** values when there are no large scans occurring, consider reducing the delay time or increasing the percentage of the buffer pool used for the old sublist. Increasing the percentage makes the old sublist larger so that it takes longer for pages in that sublist to move to the tail, which increases the likelihood that those pages are accessed again and made young. See [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion).

The **non-youngs/s** metric is applicable only to old pages. It is based on the number of page accesses. There can be multiple accesses for a given page, all of which are counted. If you do not see a higher **non-youngs/s** value when performing large table scans (and a higher **youngs/s** value), increase the delay value. See [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion).

The **young-making** rate accounts for all buffer pool page accesses, not just accesses for pages in the old sublist. The **young-making** rate and **not** rate do not normally add up to the overall buffer pool hit rate. Page hits in the old sublist cause pages to move to the new sublist, but page hits in the new sublist cause pages to move to the head of the list only if they are a certain distance from the head.

**not (young-making rate)** is the average hit rate at which page accesses have not resulted in making pages young due to the delay defined by [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) not being met, or due to page hits in the new sublist that did not result in pages being moved to the head. This rate accounts for all buffer pool page accesses, not just accesses for pages in the old sublist.

Buffer pool [server status variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables) and the [**INNODB\_BUFFER\_POOL\_STATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-pool-stats-table) table provide many of the same buffer pool metrics found in **InnoDB** Standard Monitor output. For more information, see [Example 15.10, “Querying the INNODB\_BUFFER\_POOL\_STATS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-buffer-pool-stats-example).

### 15.5.2 Change Buffer

The change buffer is a special data structure that caches changes to [secondary index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index) pages when those pages are not in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). The buffered changes, which may result from [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations (DML), are merged later when the pages are loaded into the buffer pool by other read operations.

**Figure 15.3 Change Buffer**

Unlike [clustered indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index), secondary indexes are usually nonunique, and inserts into secondary indexes happen in a relatively random order. Similarly, deletes and updates may affect secondary index pages that are not adjacently located in an index tree. Merging cached changes at a later time, when affected pages are read into the buffer pool by other operations, avoids substantial random access I/O that would be required to read secondary index pages into the buffer pool from disk.

Periodically, the purge operation that runs when the system is mostly idle, or during a slow shutdown, writes the updated index pages to disk. The purge operation can write disk blocks for a series of index values more efficiently than if each value were written to disk immediately.

Change buffer merging may take several hours when there are many affected rows and numerous secondary indexes to update. During this time, disk I/O is increased, which can cause a significant slowdown for disk-bound queries. Change buffer merging may also continue to occur after a transaction is committed, and even after a server shutdown and restart (see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery) for more information).

In memory, the change buffer occupies part of the buffer pool. On disk, the change buffer is part of the system tablespace, where index changes are buffered when the database server is shut down.

The type of data cached in the change buffer is governed by the [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) variable. For more information, see [Configuring Change Buffering](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer-configuration). You can also configure the maximum change buffer size. For more information, see [Configuring the Change Buffer Maximum Size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer-maximum-size).

Change buffering is not supported for a secondary index if the index contains a descending index column or if the primary key includes a descending index column.

For answers to frequently asked questions about the change buffer, see [Section A.16, “MySQL 8.0 FAQ: InnoDB Change Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\faqs.html#faqs-innodb-change-buffer).

#### Configuring Change Buffering

When [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations are performed on a table, the values of indexed columns (particularly the values of secondary keys) are often in an unsorted order, requiring substantial I/O to bring secondary indexes up to date. The [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer) caches changes to secondary index entries when the relevant [page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) is not in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), thus avoiding expensive I/O operations by not immediately reading in the page from disk. The buffered changes are merged when the page is loaded into the buffer pool, and the updated page is later flushed to disk. The **InnoDB** main thread merges buffered changes when the server is nearly idle, and during a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown).

Because it can result in fewer disk reads and writes, change buffering is most valuable for workloads that are I/O-bound; for example, applications with a high volume of DML operations such as bulk inserts benefit from change buffering.

However, the change buffer occupies a part of the buffer pool, reducing the memory available to cache data pages. If the working set almost fits in the buffer pool, or if your tables have relatively few secondary indexes, it may be useful to disable change buffering. If the working data set fits entirely within the buffer pool, change buffering does not impose extra overhead, because it only applies to pages that are not in the buffer pool.

The [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) variable controls the extent to which **InnoDB** performs change buffering. You can enable or disable buffering for inserts, delete operations (when index records are initially marked for deletion) and purge operations (when index records are physically deleted). An update operation is a combination of an insert and a delete. The default [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) value is **all**.

Permitted [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) values include:

***all***

The default value: buffer inserts, delete-marking operations, and purges.

***none***

Do not buffer any operations.

***inserts***

Buffer insert operations.

***deletes***

Buffer delete-marking operations.

***changes***

Buffer both inserts and delete-marking operations.

***purges***

Buffer the physical deletion operations that happen in the background.

You can set the [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) variable in the MySQL option file (**my.cnf** or **my.ini**) or change it dynamically with the [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, which requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges). Changing the setting affects the buffering of new operations; the merging of existing buffered entries is not affected.

#### Configuring the Change Buffer Maximum Size

The [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) variable permits configuring the maximum size of the change buffer as a percentage of the total size of the buffer pool. By default, [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) is set to 25. The maximum setting is 50.

Consider increasing [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) on a MySQL server with heavy insert, update, and delete activity, where change buffer merging does not keep pace with new change buffer entries, causing the change buffer to reach its maximum size limit.

Consider decreasing [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) on a MySQL server with static data used for reporting, or if the change buffer consumes too much of the memory space shared with the buffer pool, causing pages to age out of the buffer pool sooner than desired.

Test different settings with a representative workload to determine an optimal configuration. The [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) variable is dynamic, which permits modifying the setting without restarting the server.

#### Monitoring the Change Buffer

The following options are available for change buffer monitoring:

**InnoDB** Standard Monitor output includes change buffer status information. To view monitor data, issue the **SHOW ENGINE INNODB STATUS** statement.

mysql> **SHOW ENGINE INNODB STATUS\G**

Change buffer status information is located under the **INSERT BUFFER AND ADAPTIVE HASH INDEX** heading and appears similar to the following:

-------------------------------------

INSERT BUFFER AND ADAPTIVE HASH INDEX

-------------------------------------

Ibuf: size 1, free list len 0, seg size 2, 0 merges

merged operations:

insert 0, delete mark 0, delete 0

discarded operations:

insert 0, delete mark 0, delete 0

Hash table size 4425293, used cells 32, node heap has 1 buffer(s)

13577.57 hash searches/s, 202.47 non-hash searches/s

For more information, see [Section 15.17.3, “InnoDB Standard Monitor and Lock Monitor Output”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor).

The [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table provides most of the data points found in **InnoDB** Standard Monitor output plus other data points. To view change buffer metrics and a description of each, issue the following query:

mysql> **SELECT NAME, COMMENT FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME LIKE '%ibuf%'\G**

For [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table usage information, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

The [**INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) table provides metadata about each page in the buffer pool, including change buffer index and change buffer bitmap pages. Change buffer pages are identified by **PAGE\_TYPE**. **IBUF\_INDEX** is the page type for change buffer index pages, and **IBUF\_BITMAP** is the page type for change buffer bitmap pages.

**Warning**

Querying the [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) table can introduce significant performance overhead. To avoid impacting performance, reproduce the issue you want to investigate on a test instance and run your queries on the test instance.

For example, you can query the [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) table to determine the approximate number of **IBUF\_INDEX** and **IBUF\_BITMAP** pages as a percentage of total buffer pool pages.

mysql> **SELECT (SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE PAGE\_TYPE LIKE 'IBUF%') AS change\_buffer\_pages,**

**(SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE) AS total\_pages,**

**(SELECT ((change\_buffer\_pages/total\_pages)\*100))**

**AS change\_buffer\_page\_percentage;**

+---------------------+-------------+-------------------------------+

| change\_buffer\_pages | total\_pages | change\_buffer\_page\_percentage |

+---------------------+-------------+-------------------------------+

| 25 | 8192 | 0.3052 |

+---------------------+-------------+-------------------------------+

For information about other data provided by the [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) table, see [Section 26.4.2, “The INFORMATION\_SCHEMA INNODB\_BUFFER\_PAGE Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table). For related usage information, see [Section 15.15.5, “InnoDB INFORMATION\_SCHEMA Buffer Pool Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-buffer-pool-tables).

[Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html) provides change buffer mutex wait instrumentation for advanced performance monitoring. To view change buffer instrumentation, issue the following query:

mysql> **SELECT \* FROM performance\_schema.setup\_instruments**

**WHERE NAME LIKE '%wait/synch/mutex/innodb/ibuf%';**

+-------------------------------------------------------+---------+-------+

| NAME | ENABLED | TIMED |

+-------------------------------------------------------+---------+-------+

| wait/synch/mutex/innodb/ibuf\_bitmap\_mutex | YES | YES |

| wait/synch/mutex/innodb/ibuf\_mutex | YES | YES |

| wait/synch/mutex/innodb/ibuf\_pessimistic\_insert\_mutex | YES | YES |

+-------------------------------------------------------+---------+-------+

For information about monitoring **InnoDB** mutex waits, see [Section 15.16.2, “Monitoring InnoDB Mutex Waits Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-innodb-mutex-waits-performance-schema).

### 15.5.3 Adaptive Hash Index

The adaptive hash index enables **InnoDB** to perform more like an in-memory database on systems with appropriate combinations of workload and sufficient memory for the buffer pool without sacrificing transactional features or reliability. The adaptive hash index is enabled by the [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) variable, or turned off at server startup by **--skip-innodb-adaptive-hash-index**.

Based on the observed pattern of searches, a hash index is built using a prefix of the index key. The prefix can be any length, and it may be that only some values in the B-tree appear in the hash index. Hash indexes are built on demand for the pages of the index that are accessed often.

If a table fits almost entirely in main memory, a hash index speeds up queries by enabling direct lookup of any element, turning the index value into a sort of pointer. **InnoDB** has a mechanism that monitors index searches. If **InnoDB** notices that queries could benefit from building a hash index, it does so automatically.

With some workloads, the speedup from hash index lookups greatly outweighs the extra work to monitor index lookups and maintain the hash index structure. Access to the adaptive hash index can sometimes become a source of contention under heavy workloads, such as multiple concurrent joins. Queries with **LIKE** operators and **%** wildcards also tend not to benefit. For workloads that do not benefit from the adaptive hash index, turning it off reduces unnecessary performance overhead. Because it is difficult to predict in advance whether the adaptive hash index is appropriate for a particular system and workload, consider running benchmarks with it enabled and disabled.

The adaptive hash index feature is partitioned. Each index is bound to a specific partition, and each partition is protected by a separate latch. Partitioning is controlled by the [**innodb\_adaptive\_hash\_index\_parts**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index_parts) variable. The [**innodb\_adaptive\_hash\_index\_parts**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index_parts) variable is set to 8 by default. The maximum setting is 512.

You can monitor adaptive hash index use and contention in the **SEMAPHORES** section of [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output. If there are numerous threads waiting on rw-latches created in btr0sea.c, consider increasing the number of adaptive hash index partitions or disabling the adaptive hash index.

For information about the performance characteristics of hash indexes, see [Section 8.3.9, “Comparison of B-Tree and Hash Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-btree-hash).

### 15.5.4 Log Buffer

The log buffer is the memory area that holds data to be written to the log files on disk. Log buffer size is defined by the [**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) variable. The default size is 16MB. The contents of the log buffer are periodically flushed to disk. A large log buffer enables large transactions to run without the need to write redo log data to disk before the transactions commit. Thus, if you have transactions that update, insert, or delete many rows, increasing the size of the log buffer saves disk I/O.

The [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) variable controls how the contents of the log buffer are written and flushed to disk. The [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout) variable controls log flushing frequency.

For related information, see [Memory Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-memory-configuration), and [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

## 15.6 InnoDB On-Disk Structures

[15.6.1 Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tables)

[15.6.2 Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-indexes)

[15.6.3 Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace)

[15.6.4 Doublewrite Buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer)

[15.6.5 Redo Log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-redo-log)

[15.6.6 Undo Logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-logs)

This section describes **InnoDB** on-disk structures and related topics.

### 15.6.1 Tables

[15.6.1.1 Creating InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#using-innodb-tables)

[15.6.1.2 Creating Tables Externally](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external)

[15.6.1.3 Importing InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import)

[15.6.1.4 Moving or Copying InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-migration)

[15.6.1.5 Converting Tables from MyISAM to InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#converting-tables-to-innodb)

[15.6.1.6 AUTO\_INCREMENT Handling in InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling)

This section covers topics related to **InnoDB** tables.

#### 15.6.1.1 Creating InnoDB Tables

**InnoDB** tables are created using the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement; for example:

CREATE TABLE t1 (a INT, b CHAR (20), PRIMARY KEY (a)) ENGINE=InnoDB;

The **ENGINE=InnoDB** clause is not required when **InnoDB** is defined as the default storage engine, which it is by default. However, the **ENGINE** clause is useful if the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement is to be replayed on a different MySQL Server instance where the default storage engine is not **InnoDB** or is unknown. You can determine the default storage engine on a MySQL Server instance by issuing the following statement:

mysql> **SELECT @@default\_storage\_engine;**

+--------------------------+

| @@default\_storage\_engine |

+--------------------------+

| InnoDB |

+--------------------------+

**InnoDB** tables are created in file-per-table tablespaces by default. To create an **InnoDB** table in the **InnoDB** system tablespace, disable the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable before creating the table. To create an **InnoDB** table in a general tablespace, use [**CREATE TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) syntax. For more information, see [Section 15.6.3, “Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace).

##### Row Formats

The row format of an **InnoDB** table determines how its rows are physically stored on disk. **InnoDB** supports four row formats, each with different storage characteristics. Supported row formats include **REDUNDANT**, **COMPACT**, **DYNAMIC**, and **COMPRESSED**. The **DYNAMIC** row format is the default. For information about row format characteristics, see [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

The [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable defines the default row format. The row format of a table can also be defined explicitly using the **ROW\_FORMAT** table option in a **CREATE TABLE** or **ALTER TABLE** statement. See [Defining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-defining).

##### Primary Keys

It is recommended that you define a primary key for each table that you create. When selecting primary key columns, choose columns with the following characteristics:

Columns that are referenced by the most important queries.

Columns that are never left blank.

Columns that never have duplicate values.

Columns that rarely if ever change value once inserted.

For example, in a table containing information about people, you would not create a primary key on **(firstname, lastname)** because more than one person can have the same name, a name column may be left blank, and sometimes people change their names. With so many constraints, often there is not an obvious set of columns to use as a primary key, so you create a new column with a numeric ID to serve as all or part of the primary key. You can declare an [auto-increment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_auto_increment) column so that ascending values are filled in automatically as rows are inserted:

# The value of ID can act like a pointer between related items in different tables.

CREATE TABLE t5 (id INT AUTO\_INCREMENT, b CHAR (20), PRIMARY KEY (id));

# The primary key can consist of more than one column. Any autoinc column must come first.

CREATE TABLE t6 (id INT AUTO\_INCREMENT, a INT, b CHAR (20), PRIMARY KEY (id,a));

For more information about auto-increment columns, see [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling).

Although a table works correctly without defining a primary key, the primary key is involved with many aspects of performance and is a crucial design aspect for any large or frequently used table. It is recommended that you always specify a primary key in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. If you create the table, load data, and then run [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to add a primary key later, that operation is much slower than defining the primary key when creating the table. For more information about primary keys, see [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types).

##### Viewing InnoDB Table Properties

To view the properties of an **InnoDB** table, issue a [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) statement:

mysql> **SHOW TABLE STATUS FROM test LIKE 't%' \G;**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: t1

Engine: InnoDB

Version: 10

Row\_format: Dynamic

Rows: 0

Avg\_row\_length: 0

Data\_length: 16384

Max\_data\_length: 0

Index\_length: 0

Data\_free: 0

Auto\_increment: NULL

Create\_time: 2021-02-18 12:18:28

Update\_time: NULL

Check\_time: NULL

Collation: utf8mb4\_0900\_ai\_ci

Checksum: NULL

Create\_options:

Comment:

For information about [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) output, see [Section 13.7.7.38, “SHOW TABLE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status).

You can also access **InnoDB** table properties by querying the **InnoDB** Information Schema system tables:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLES WHERE NAME='test/t1' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 1144

NAME: test/t1

FLAG: 33

N\_COLS: 5

SPACE: 30

ROW\_FORMAT: Dynamic

ZIP\_PAGE\_SIZE: 0

SPACE\_TYPE: Single

INSTANT\_COLS: 0

For more information, see [Section 15.15.3, “InnoDB INFORMATION\_SCHEMA Schema Object Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-system-tables).

#### 15.6.1.2 Creating Tables Externally

There are different reasons for creating **InnoDB** tables externally; that is, creating tables outside of the data directory. Those reasons might include space management, I/O optimization, or placing tables on a storage device with particular performance or capacity characteristics, for example.

**InnoDB** supports the following methods for creating tables externally:

[Using the DATA DIRECTORY Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-data-directory)

[Using CREATE TABLE ... TABLESPACE Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-tablespace-syntax)

[Creating a Table in an External General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-tablespace)

##### Using the DATA DIRECTORY Clause

You can create an **InnoDB** table in an external directory by specifying a **DATA DIRECTORY** clause in the **CREATE TABLE** statement.

CREATE TABLE t1 (c1 INT PRIMARY KEY) DATA DIRECTORY = '***/external/directory***';

The **DATA DIRECTORY** clause is supported for tables created in file-per-table tablespaces. Tables are implicitly created in file-per-table tablespaces when the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable is enabled, which it is by default.

mysql> **SELECT @@innodb\_file\_per\_table;**

+-------------------------+

| @@innodb\_file\_per\_table |

+-------------------------+

| 1 |

+-------------------------+

For more information about file-per-table tablespaces, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces).

When you specify a **DATA DIRECTORY** clause in a **CREATE TABLE** statement, the table's data file (***table\_name***.ibd) is created in a schema directory under the specified directory.

As of MySQL 8.0.21, tables and table partitions created outside of the data directory using the **DATA DIRECTORY** clause are restricted to directories known to **InnoDB**. This requirement permits database administrators to control where tablespace data files are created and ensures that data files can be found during recovery (see [Tablespace Discovery During Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-tablespace-discovery)). Known directories are those defined by the [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), and [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variables. You can use the following statement to check those settings:

mysql> SELECT @@datadir,@@innodb\_data\_home\_dir,@@innodb\_directories;

If the directory you want to use is unknown, add it to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting before you create the table. The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable is read-only. Configuring it requires restarting the server. For general information about setting system variables, see [Section 5.1.9, “Using System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#using-system-variables).

The following example demonstrates creating a table in an external directory using the **DATA DIRECTORY** clause. It is assumed that the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable is enabled and that the directory is known to **InnoDB**.

mysql> **USE test;**

Database changed

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) DATA DIRECTORY = '*/external/directory*';**

# MySQL creates the table's data file in a schema directory

# under the external directory

shell> **cd /external/directory/test**

shell> **ls**

t1.ibd

###### Usage Notes:

MySQL initially holds the tablespace data file open, preventing you from dismounting the device, but might eventually close the file if the server is busy. Be careful not to accidentally dismount an external device while MySQL is running, or start MySQL while the device is disconnected. Attempting to access a table when the associated data file is missing causes a serious error that requires a server restart.

A server restart might fail if the data file is not found at the expected path. In this case, you can restore the tablespace data file from a backup or drop the table to remove the information about it from the [data dictionary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_dictionary).

Before placing a table on an NFS-mounted volume, review potential issues outlined in [Using NFS with MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#disk-issues-nfs).

If using an LVM snapshot, file copy, or other file-based mechanism to back up the table's data file, always use the [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) statement first to ensure that all changes buffered in memory are [flushed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) to disk before the backup occurs.

Using the **DATA DIRECTORY** clause to create a table in an external directory is an alternative to using [symbolic links](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#symbolic-links), which **InnoDB** does not support.

The **DATA DIRECTORY** clause is not supported in a replication environment where the source and replica reside on the same host. The **DATA DIRECTORY** clause requires a full directory path. Replicating the path in this case would cause the source and replica to create the table in same location.

As of MySQL 8.0.21, tables created in file-per-table tablespaces can no longer be created in the undo tablespace directory ([**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory)) unless that directly is known to **InnoDB**. Known directories are those defined by the [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), and [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variables.

##### Using CREATE TABLE ... TABLESPACE Syntax

[**CREATE TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) syntax can be used in combination with the **DATA DIRECTORY** clause to create a table in an external directory. To do so, specify **innodb\_file\_per\_table** as the tablespace name.

mysql> **CREATE TABLE t2 (c1 INT PRIMARY KEY) TABLESPACE = innodb\_file\_per\_table**

**DATA DIRECTORY = '/external/directory';**

This method is supported only for tables created in file-per-table tablespaces, but does not require the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable to be enabled. In all other respects, this method is equivalent to the **CREATE TABLE ... DATA DIRECTORY** method described above. The same usage notes apply.

##### Creating a Table in an External General Tablespace

You can create a table in a general tablespace that resides in an external directory.

For information about creating a general tablespace in an external directory, see [Creating a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-creating).

For information about creating a table in a general tablespace, see [Adding Tables to a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-adding-tables).

#### 15.6.1.3 Importing InnoDB Tables

This section describes how to import tables using the Transportable Tablespaces feature, which permits importing tables, partitioned tables, or individual table partitions that reside in file-per-table tablespaces. There are many reasons why you might want to import tables:

To run reports on a non-production MySQL server instance to avoid placing extra load on a production server.

To copy data to a new replica server.

To restore a table from a backed-up tablespace file.

As a faster way of moving data than importing a dump file, which requires reinserting data and rebuilding indexes.

To move a data to a server with storage media that is better suited to your storage requirements. For example, you might move busy tables to an SSD device, or move large tables to a high-capacity HDD device.

The Transportable Tablespaces feature is described under the following topics in this section:

[Prerequisites](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-prerequsites)

[Importing Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-example)

[Importing Partitioned Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-partitioned-table)

[Importing Table Partitions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-partitions)

[Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-limitations)

[Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-usage-notes)

[Internals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import-internals)

##### Prerequisites

The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable must be enabled, which it is by default.

The page size of the tablespace must match the page size of the destination MySQL server instance. **InnoDB** page size is defined by the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) variable, which is configured when initializing a MySQL server instance.

If the table has a foreign key relationship, [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) must be disabled before executing **DISCARD TABLESPACE**. Also, you should export all foreign key related tables at the same logical point in time, as [**ALTER TABLE ... IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) does not enforce foreign key constraints on imported data. To do so, stop updating the related tables, commit all transactions, acquire shared locks on the tables, and perform the export operations.

When importing a table from another MySQL server instance, both MySQL server instances must have General Availability (GA) status and must be the same version. Otherwise, the table must be created on the same MySQL server instance into which it is being imported.

If the table was created in an external directory by specifying the **DATA DIRECTORY** clause in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, the table that you replace on the destination instance must be defined with the same **DATA DIRECTORY** clause. A schema mismatch error is reported if the clauses do not match. To determine if the source table was defined with a **DATA DIRECTORY** clause, use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) to view the table definition. For information about using the **DATA DIRECTORY** clause, see [Section 15.6.1.2, “Creating Tables Externally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external).

If a **ROW\_FORMAT** option is not defined explicitly in the table definition or **ROW\_FORMAT=DEFAULT** is used, the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) setting must be the same on the source and destination instances. Otherwise, a schema mismatch error is reported when you attempt the import operation. Use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) to check the table definition. Use [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) to check the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) setting. For related information, see [Defining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-defining).

##### Importing Tables

This example demonstrates how to import a regular non-partitioned table that resides in a file-per-table tablespace.

On the destination instance, create a table with the same definition as the table you intend to import. (You can obtain the table definition using [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> USE test;

mysql> CREATE TABLE t1 (c1 INT) ENGINE=INNODB;

On the destination instance, discard the tablespace of the table that you just created. (Before importing, you must discard the tablespace of the receiving table.)

mysql> ALTER TABLE t1 DISCARD TABLESPACE;

On the source instance, run [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) to quiesce the table you intend to import. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> USE test;

mysql> FLUSH TABLES t1 FOR EXPORT;

[**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) ensures that changes to the named table are flushed to disk so that a binary table copy can be made while the server is running. When [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is run, **InnoDB** generates a .cfg metadata file in the schema directory of the table. The .cfg file contains metadata that is used for schema verification during the import operation.

Copy the .ibd file and .cfg metadata file from the source instance to the destination instance. For example:

shell> scp ***/path/to/datadir***/test/t1.{ibd,cfg} destination-server:***/path/to/datadir***/test

The .ibd file and .cfg file must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing a table from an encrypted tablespace, **InnoDB** generates a .cfp file in addition to a .cfg metadata file. The .cfp file must be copied to the destination instance together with the .cfg file. The .cfp file contains a transfer key and an encrypted tablespace key. On import, **InnoDB** uses the transfer key to decrypt the tablespace key. For related information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

On the source instance, use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks acquired by the [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) statement:

mysql> USE test;

mysql> UNLOCK TABLES;

On the destination instance, import the tablespace:

mysql> USE test;

mysql> ALTER TABLE t1 IMPORT TABLESPACE;

##### Importing Partitioned Tables

This example demonstrates how to import a partitioned table, where each table partition resides in a file-per-table tablespace.

On the destination instance, create a partitioned table with the same definition as the partitioned table that you want to import. (You can obtain the table definition using [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> **USE test;**

mysql> **CREATE TABLE t1 (i int) ENGINE = InnoDB PARTITION BY KEY (i) PARTITIONS 3;**

In the **/*datadir*/test** directory, there is a tablespace .ibd file for each of the three partitions.

mysql> **\! ls */path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd

On the destination instance, discard the tablespace for the partitioned table. (Before the import operation, you must discard the tablespace of the receiving table.)

mysql> **ALTER TABLE t1 DISCARD TABLESPACE;**

The three tablespace .ibd files of the partitioned table are discarded from the **/*datadir*/test** directory.

On the source instance, run [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) to quiesce the partitioned table that you intend to import. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> **USE test;**

mysql> **FLUSH TABLES t1 FOR EXPORT;**

[**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) ensures that changes to the named table are flushed to disk so that binary table copy can be made while the server is running. When [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is run, **InnoDB** generates .cfg metadata files in the schema directory of the table for each of the table's tablespace files.

mysql> **\! ls */path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd

t1#p#p0.cfg t1#p#p1.cfg t1#p#p2.cfg

The .cfg files contain metadata that is used for schema verification when importing the tablespace. [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) can only be run on the table, not on individual table partitions.

Copy the .ibd and .cfg files from the source instance schema directory to the destination instance schema directory. For example:

shell>scp ***/path/to/datadir*/test/t1\*.{ibd,cfg} destination-server:*/path/to/datadir*/test**

The .ibd and .cfg files must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing a table from an encrypted tablespace, **InnoDB** generates a .cfp files in addition to a .cfg metadata files. The .cfp files must be copied to the destination instance together with the .cfg files. The .cfp files contain a transfer key and an encrypted tablespace key. On import, **InnoDB** uses the transfer key to decrypt the tablespace key. For related information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

On the source instance, use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks acquired by [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list):

mysql> **USE test;**

mysql> **UNLOCK TABLES;**

On the destination instance, import the tablespace of the partitioned table:

mysql> **USE test;**

mysql> **ALTER TABLE t1 IMPORT TABLESPACE;**

##### Importing Table Partitions

This example demonstrates how to import individual table partitions, where each partition resides in a file-per-table tablespace file.

In the following example, two partitions (**p2** and **p3**) of a four-partition table are imported.

On the destination instance, create a partitioned table with the same definition as the partitioned table that you want to import partitions from. (You can obtain the table definition using [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) syntax.) If the table definition does not match, a schema mismatch error is reported when you attempt the import operation.

mysql> **USE test;**

mysql> **CREATE TABLE t1 (i int) ENGINE = InnoDB PARTITION BY KEY (i) PARTITIONS 4;**

In the **/*datadir*/test** directory, there is a tablespace .ibd file for each of the four partitions.

mysql> **\! ls */path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd t1#p#p3.ibd

On the destination instance, discard the partitions that you intend to import from the source instance. (Before importing partitions, you must discard the corresponding partitions from the receiving partitioned table.)

mysql> **ALTER TABLE t1 DISCARD PARTITION p2, p3 TABLESPACE;**

The tablespace **.ibd** files for the two discarded partitions are removed from the **/*datadir*/test** directory on the destination instance, leaving the following files:

mysql> **\! ls */path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd

**Note**

When [**ALTER TABLE ... DISCARD PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is run on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.

On the source instance, run [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) to quiesce the partitioned table. When a table is quiesced, only read-only transactions are permitted on the table.

mysql> **USE test;**

mysql> **FLUSH TABLES t1 FOR EXPORT;**

[**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) ensures that changes to the named table are flushed to disk so that binary table copy can be made while the instance is running. When [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is run, **InnoDB** generates a .cfg metadata file for each of the table's tablespace files in the schema directory of the table.

mysql> **\! ls */path/to/datadir*/test/**

t1#p#p0.ibd t1#p#p1.ibd t1#p#p2.ibd t1#p#p3.ibd

t1#p#p0.cfg t1#p#p1.cfg t1#p#p2.cfg t1#p#p3.cfg

The .cfg files contain metadata that used for schema verification during the import operation. [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) can only be run on the table, not on individual table partitions.

Copy the .ibd and .cfg files for partition **p2** and partition **p3** from the source instance schema directory to the destination instance schema directory.

shell> **scp t1#p#p2.ibd t1#p#p2.cfg t1#p#p3.ibd t1#p#p3.cfg destination-server:*/path/to/datadir*/test**

The .ibd and .cfg files must be copied before releasing the shared locks, as described in the next step.

**Note**

If you are importing partitions from an encrypted tablespace, **InnoDB** generates a .cfp files in addition to a .cfg metadata files. The .cfp files must be copied to the destination instance together with the .cfg files. The .cfp files contain a transfer key and an encrypted tablespace key. On import, **InnoDB** uses the transfer key to decrypt the tablespace key. For related information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

On the source instance, use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks acquired by [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list):

mysql> **USE test;**

mysql> **UNLOCK TABLES;**

On the destination instance, import table partitions **p2** and **p3**:

mysql> **USE test;**

mysql> **ALTER TABLE t1 IMPORT PARTITION p2, p3 TABLESPACE;**

**Note**

When [**ALTER TABLE ... IMPORT PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is run on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.

##### Limitations

The Transportable Tablespaces feature is only supported for tables that reside in file-per-table tablespaces. It is not supported for the tables that reside in the system tablespace or general tablespaces. Tables in shared tablespaces cannot be quiesced.

[**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is not supported on tables with a **FULLTEXT** index, as full-text search auxiliary tables cannot be flushed. After importing a table with a **FULLTEXT** index, run [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) to rebuild the **FULLTEXT** indexes. Alternatively, drop **FULLTEXT** indexes before the export operation and recreate the indexes after importing the table on the destination instance.

Due to a **.cfg** metadata file limitation, schema mismatches are not reported for partition type or partition definition differences when importing a partitioned table. Column differences are reported.

Prior to MySQL 8.0.19, index key part sort order information is not stored to the **.cfg** metadata file used during a tablespace import operation. The index key part sort order is therefore assumed to be ascending, which is the default. As a result, records could be sorted in an unintended order if one table involved in the import operation is defined with a DESC index key part sort order and the other table is not. The workaround is to drop and recreate affected indexes. For information about index key part sort order, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

The **.cfg** file format was updated in MySQL 8.0.19 to include index key part sort order information. The issue described above does not affect import operations between MySQL 8.0.19 server instances or higher.

##### Usage Notes

[**ALTER TABLE ... IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) does not require a .cfg metadata file to import a table. However, metadata checks are not performed when importing without a .cfg file, and a warning similar to the following is issued:

Message: InnoDB: IO Read error: (2, No such file or directory) Error opening '.\

test\t.cfg', will attempt to import without schema verification

1 row in set (0.00 sec)

Importing a table without a .cfg metadata file should only be considered if no schema mismatches are expected. The ability to import without a .cfg file could be useful in crash recovery scenarios where metadata is not accessible.

On Windows, **InnoDB** stores database, tablespace, and table names internally in lowercase. To avoid import problems on case-sensitive operating systems such as Linux and Unix, create all databases, tablespaces, and tables using lowercase names. A convenient way to ensure that names are created in lowercase is to set [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) to 1 before initializing the server. (It is prohibited to start the server with a [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting that is different from the setting used when the server was initialized.)

[mysqld]

lower\_case\_table\_names=1

When running [**ALTER TABLE ... DISCARD PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**ALTER TABLE ... IMPORT PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on subpartitioned tables, both partition and subpartition table names are permitted. When a partition name is specified, subpartitions of that partition are included in the operation.

##### Internals

The following information describes internals and messages written to the error log during a table import procedure.

When [**ALTER TABLE ... DISCARD TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is run on the destination instance:

The table is locked in X mode.

The tablespace is detached from the table.

When [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is run on the source instance:

The table being flushed for export is locked in shared mode.

The purge coordinator thread is stopped.

Dirty pages are synchronized to disk.

Table metadata is written to the binary .cfg file.

Expected error log messages for this operation:

[Note] InnoDB: Sync to disk of '"test"."t1"' started.

[Note] InnoDB: Stopping purge

[Note] InnoDB: Writing table metadata to './test/t1.cfg'

[Note] InnoDB: Table '"test"."t1"' flushed to disk

When [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is run on the source instance:

The binary **.cfg** file is deleted.

The shared lock on the table or tables being imported is released and the purge coordinator thread is restarted.

Expected error log messages for this operation:

[Note] InnoDB: Deleting the meta-data file './test/t1.cfg'

[Note] InnoDB: Resuming purge

When [**ALTER TABLE ... IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is run on the destination instance, the import algorithm performs the following operations for each tablespace being imported:

Each tablespace page is checked for corruption.

The space ID and log sequence numbers (LSNs) on each page are updated.

Flags are validated and LSN updated for the header page.

Btree pages are updated.

The page state is set to dirty so that it is written to disk.

Expected error log messages for this operation:

[Note] InnoDB: Importing tablespace for table 'test/t1' that was exported

from host '***host\_name***'

[Note] InnoDB: Phase I - Update all pages

[Note] InnoDB: Sync to disk

[Note] InnoDB: Sync to disk - done!

[Note] InnoDB: Phase III - Flush changes to disk

[Note] InnoDB: Phase IV - Flush complete

**Note**

You may also receive a warning that a tablespace is discarded (if you discarded the tablespace for the destination table) and a message stating that statistics could not be calculated due to a missing .ibd file:

[Warning] InnoDB: Table "test"."t1" tablespace is set as discarded.

7f34d9a37700 InnoDB: cannot calculate statistics for table

"test"."t1" because the .ibd file is missing. For help, please refer to

http://dev.mysql.com/doc/refman/8.0/en/innodb-troubleshooting.html

#### 15.6.1.4 Moving or Copying InnoDB Tables

This section describes techniques for moving or copying some or all **InnoDB** tables to a different server or instance. For example, you might move an entire MySQL instance to a larger, faster server; you might clone an entire MySQL instance to a new replica server; you might copy individual tables to another instance to develop and test an application, or to a data warehouse server to produce reports.

On Windows, **InnoDB** always stores database and table names internally in lowercase. To move databases in a binary format from Unix to Windows or from Windows to Unix, create all databases and tables using lowercase names. A convenient way to accomplish this is to add the following line to the **[mysqld]** section of your my.cnf or my.ini file before creating any databases or tables:

[mysqld]

lower\_case\_table\_names=1

**Note**

It is prohibited to start the server with a [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting that is different from the setting used when the server was initialized.

Techniques for moving or copying **InnoDB** tables include:

[Importing Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#copy-tables-import)

[MySQL Enterprise Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#copy-tables-meb)

[Copying Data Files (Cold Backup Method)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#copy-tables-cold-backup)

[Restoring from a Logical Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#copy-tables-logical-backup)

##### Importing Tables

A table that resides in a file-per-table tablespace can be imported from another MySQL server instance or from a backup using the Transportable Tablespace feature. See [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

##### MySQL Enterprise Backup

The MySQL Enterprise Backup product lets you back up a running MySQL database with minimal disruption to operations while producing a consistent snapshot of the database. When MySQL Enterprise Backup is copying tables, reads and writes can continue. In addition, MySQL Enterprise Backup can create compressed backup files, and back up subsets of tables. In conjunction with the MySQL binary log, you can perform point-in-time recovery. MySQL Enterprise Backup is included as part of the MySQL Enterprise subscription.

For more details about MySQL Enterprise Backup, see [Section 30.2, “MySQL Enterprise Backup Overview”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup).

##### Copying Data Files (Cold Backup Method)

You can move an **InnoDB** database simply by copying all the relevant files listed under "Cold Backups" in [Section 15.18.1, “InnoDB Backup”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-backup).

**InnoDB** data and log files are binary-compatible on all platforms having the same floating-point number format. If the floating-point formats differ but you have not used [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) or [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) data types in your tables, then the procedure is the same: simply copy the relevant files.

When you move or copy file-per-table .ibd files, the database directory name must be the same on the source and destination systems. The table definition stored in the **InnoDB** shared tablespace includes the database name. The transaction IDs and log sequence numbers stored in the tablespace files also differ between databases.

To move an .ibd file and the associated table from one database to another, use a [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) statement:

RENAME TABLE ***db1.tbl\_name*** TO ***db2.tbl\_name***;

If you have a “clean” backup of an .ibd file, you can restore it to the MySQL installation from which it originated as follows:

The table must not have been dropped or truncated since you copied the .ibd file, because doing so changes the table ID stored inside the tablespace.

Issue this [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to delete the current .ibd file:

ALTER TABLE ***tbl\_name*** DISCARD TABLESPACE;

Copy the backup .ibd file to the proper database directory.

Issue this [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to tell **InnoDB** to use the new .ibd file for the table:

ALTER TABLE ***tbl\_name*** IMPORT TABLESPACE;

**Note**

The [**ALTER TABLE ... IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) feature does not enforce foreign key constraints on imported data.

In this context, a “clean” .ibd file backup is one for which the following requirements are satisfied:

There are no uncommitted modifications by transactions in the .ibd file.

There are no unmerged insert buffer entries in the .ibd file.

Purge has removed all delete-marked index records from the .ibd file.

[**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) has flushed all modified pages of the .ibd file from the buffer pool to the file.

You can make a clean backup .ibd file using the following method:

Stop all activity from the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) server and commit all transactions.

Wait until [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) shows that there are no active transactions in the database, and the main thread status of **InnoDB** is **Waiting for server activity**. Then you can make a copy of the .ibd file.

Another method for making a clean copy of an .ibd file is to use the MySQL Enterprise Backup product:

Use MySQL Enterprise Backup to back up the **InnoDB** installation.

Start a second [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) server on the backup and let it clean up the .ibd files in the backup.

##### Restoring from a Logical Backup

You can use a utility such as [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to perform a logical backup, which produces a set of SQL statements that can be executed to reproduce the original database object definitions and table data for transfer to another SQL server. Using this method, it does not matter whether the formats differ or if your tables contain floating-point data.

To improve the performance of this method, disable [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) when importing data. Perform a commit only after importing an entire table or segment of a table.

#### 15.6.1.5 Converting Tables from MyISAM to InnoDB

If you have [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables that you want to convert to [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) for better reliability and scalability, review the following guidelines and tips before converting.

**Note**

Partitioned **MyISAM** tables created in previous versions of MySQL are not compatible with MySQL 8.0. Such tables must be prepared prior to upgrade, either by removing the partitioning, or by converting them to **InnoDB**. See [Section 24.6.2, “Partitioning Limitations Relating to Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-storage-engines), for more information.

[Adjusting Memory Usage for MyISAM and InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-memory-usage)

[Handling Too-Long Or Too-Short Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-transactions)

[Handling Deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-deadlock)

[Storage Layout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-plan-storage)

[Converting an Existing Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-convert)

[Cloning the Structure of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-clone)

[Transferring Data](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-transfer)

[Storage Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-storage-requirements)

[Defining Primary Keys](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-primary-key)

[Application Performance Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-application-performance)

[Understanding Files Associated with InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-convert-understand-files)

##### Adjusting Memory Usage for MyISAM and InnoDB

As you transition away from **MyISAM** tables, lower the value of the [**key\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_key_buffer_size) configuration option to free memory no longer needed for caching results. Increase the value of the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) configuration option, which performs a similar role of allocating cache memory for **InnoDB** tables. The **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) caches both table data and index data, speeding up lookups for queries and keeping query results in memory for reuse. For guidance regarding buffer pool size configuration, see [Section 8.12.3.1, “How MySQL Uses Memory”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#memory-use).

##### Handling Too-Long Or Too-Short Transactions

Because **MyISAM** tables do not support [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction), you might not have paid much attention to the [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) configuration option and the [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) and [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statements. These keywords are important to allow multiple sessions to read and write **InnoDB** tables concurrently, providing substantial scalability benefits in write-heavy workloads.

While a transaction is open, the system keeps a snapshot of the data as seen at the beginning of the transaction, which can cause substantial overhead if the system inserts, updates, and deletes millions of rows while a stray transaction keeps running. Thus, take care to avoid transactions that run for too long:

If you are using a [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) session for interactive experiments, always [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) (to finalize the changes) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) (to undo the changes) when finished. Close down interactive sessions rather than leave them open for long periods, to avoid keeping transactions open for long periods by accident.

Make sure that any error handlers in your application also [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) incomplete changes or [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) completed changes.

[**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) is a relatively expensive operation, because [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations are written to **InnoDB** tables prior to the [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), with the expectation that most changes are committed successfully and rollbacks are rare. When experimenting with large volumes of data, avoid making changes to large numbers of rows and then rolling back those changes.

When loading large volumes of data with a sequence of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements, periodically [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) the results to avoid having transactions that last for hours. In typical load operations for data warehousing, if something goes wrong, you truncate the table (using [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table)) and start over from the beginning rather than doing a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

The preceding tips save memory and disk space that can be wasted during too-long transactions. When transactions are shorter than they should be, the problem is excessive I/O. With each [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), MySQL makes sure each change is safely recorded to disk, which involves some I/O.

For most operations on **InnoDB** tables, you should use the setting [**autocommit=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit). From an efficiency perspective, this avoids unnecessary I/O when you issue large numbers of consecutive [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements. From a safety perspective, this allows you to issue a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement to recover lost or garbled data if you make a mistake on the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) command line, or in an exception handler in your application.

[**autocommit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is suitable for **InnoDB** tables when running a sequence of queries for generating reports or analyzing statistics. In this situation, there is no I/O penalty related to [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), and **InnoDB** can [automatically optimize the read-only workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#innodb-performance-ro-txn).

If you make a series of related changes, finalize all the changes at once with a single [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) at the end. For example, if you insert related pieces of information into several tables, do a single [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) after making all the changes. Or if you run many consecutive [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements, do a single [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) after all the data is loaded; if you are doing millions of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements, perhaps split up the huge transaction by issuing a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) every ten thousand or hundred thousand records, so the transaction does not grow too large.

Remember that even a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement opens a transaction, so after running some report or debugging queries in an interactive [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) session, either issue a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or close the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) session.

For related information, see [Section 15.7.2.2, “autocommit, Commit, and Rollback”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-autocommit-commit-rollback).

##### Handling Deadlocks

You might see warning messages referring to “deadlocks” in the MySQL error log, or the output of [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine). A [deadlock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) is not a serious issue for **InnoDB** tables, and often does not require any corrective action. When two transactions start modifying multiple tables, accessing the tables in a different order, they can reach a state where each transaction is waiting for the other and neither can proceed. When [deadlock detection](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock_detection) is enabled (the default), MySQL immediately detects this condition and cancels ([rolls back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback)) the “smaller” transaction, allowing the other to proceed. If deadlock detection is disabled using the [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) configuration option, **InnoDB** relies on the [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) setting to roll back transactions in case of a deadlock.

Either way, your applications need error-handling logic to restart a transaction that is forcibly cancelled due to a deadlock. When you re-issue the same SQL statements as before, the original timing issue no longer applies. Either the other transaction has already finished and yours can proceed, or the other transaction is still in progress and your transaction waits until it finishes.

If deadlock warnings occur constantly, you might review the application code to reorder the SQL operations in a consistent way, or to shorten the transactions. You can test with the [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) option enabled to see all deadlock warnings in the MySQL error log, rather than only the last warning in the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output.

For more information, see [Section 15.7.5, “Deadlocks in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks).

##### Storage Layout

To get the best performance from **InnoDB** tables, you can adjust a number of parameters related to storage layout.

When you convert **MyISAM** tables that are large, frequently accessed, and hold vital data, investigate and consider the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) and [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) variables, and the [**ROW\_FORMAT** and **KEY\_BLOCK\_SIZE** clauses](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format) of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

During your initial experiments, the most important setting is [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table). When this setting is enabled, which is the default, new **InnoDB** tables are implicitly created in [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces. In contrast with the **InnoDB** system tablespace, file-per-table tablespaces allow disk space to be reclaimed by the operating system when a table is truncated or dropped. File-per-table tablespaces also support [DYNAMIC](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dynamic_row_format) and [COMPRESSED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compressed_row_format) row formats and associated features such as table compression, efficient off-page storage for long variable-length columns, and large index prefixes. For more information, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces).

You can also store **InnoDB** tables in a shared general tablespace, which support multiple tables and all row formats. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

##### Converting an Existing Table

To convert a non-**InnoDB** table to use **InnoDB** use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

ALTER TABLE ***table\_name*** ENGINE=InnoDB;

##### Cloning the Structure of a Table

You might make an **InnoDB** table that is a clone of a MyISAM table, rather than using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to perform conversion, to test the old and new table side-by-side before switching.

Create an empty **InnoDB** table with identical column and index definitions. Use **SHOW CREATE TABLE *table\_name*\G** to see the full [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement to use. Change the **ENGINE** clause to **ENGINE=INNODB**.

##### Transferring Data

To transfer a large volume of data into an empty **InnoDB** table created as shown in the previous section, insert the rows with **INSERT INTO *innodb\_table* SELECT \* FROM *myisam\_table* ORDER BY *primary\_key\_columns***.

You can also create the indexes for the **InnoDB** table after inserting the data. Historically, creating new secondary indexes was a slow operation for **InnoDB**, but now you can create the indexes after the data is loaded with relatively little overhead from the index creation step.

If you have **UNIQUE** constraints on secondary keys, you can speed up a table import by turning off the uniqueness checks temporarily during the import operation:

SET unique\_checks=0;

***... import operation ...***

SET unique\_checks=1;

For big tables, this saves disk I/O because **InnoDB** can use its [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer) to write secondary index records as a batch. Be certain that the data contains no duplicate keys. [**unique\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_unique_checks) permits but does not require storage engines to ignore duplicate keys.

For better control over the insertion process, you can insert big tables in pieces:

INSERT INTO newtable SELECT \* FROM oldtable

WHERE yourkey > ***something*** AND yourkey <= ***somethingelse***;

After all records are inserted, you can rename the tables.

During the conversion of big tables, increase the size of the **InnoDB** buffer pool to reduce disk I/O. Typically, the recommended buffer pool size is 50 to 75 percent of system memory. You can also increase the size of **InnoDB** log files.

##### Storage Requirements

If you intend to make several temporary copies of your data in **InnoDB** tables during the conversion process, it is recommended that you create the tables in file-per-table tablespaces so that you can reclaim the disk space when you drop the tables. When the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) configuration option is enabled (the default), newly created **InnoDB** tables are implicitly created in file-per-table tablespaces.

Whether you convert the **MyISAM** table directly or create a cloned **InnoDB** table, make sure that you have sufficient disk space to hold both the old and new tables during the process. ***InnoDB tables require more disk space than MyISAM tables.*** If an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation runs out of space, it starts a rollback, and that can take hours if it is disk-bound. For inserts, **InnoDB** uses the insert buffer to merge secondary index records to indexes in batches. That saves a lot of disk I/O. For rollback, no such mechanism is used, and the rollback can take 30 times longer than the insertion.

In the case of a runaway rollback, if you do not have valuable data in your database, it may be advisable to kill the database process rather than wait for millions of disk I/O operations to complete. For the complete procedure, see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

##### Defining Primary Keys

The **PRIMARY KEY** clause is a critical factor affecting the performance of MySQL queries and the space usage for tables and indexes. The primary key uniquely identifies a row in a table. Every row in the table should have a primary key value, and no two rows can have the same primary key value.

These are guidelines for the primary key, followed by more detailed explanations.

Declare a **PRIMARY KEY** for each table. Typically, it is the most important column that you refer to in **WHERE** clauses when looking up a single row.

Declare the **PRIMARY KEY** clause in the original [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, rather than adding it later through an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement.

Choose the column and its data type carefully. Prefer numeric columns over character or string ones.

Consider using an auto-increment column if there is not another stable, unique, non-null, numeric column to use.

An auto-increment column is also a good choice if there is any doubt whether the value of the primary key column could ever change. Changing the value of a primary key column is an expensive operation, possibly involving rearranging data within the table and within each secondary index.

Consider adding a [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) to any table that does not already have one. Use the smallest practical numeric type based on the maximum projected size of the table. This can make each row slightly more compact, which can yield substantial space savings for large tables. The space savings are multiplied if the table has any [secondary indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index), because the primary key value is repeated in each secondary index entry. In addition to reducing data size on disk, a small primary key also lets more data fit into the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), speeding up all kinds of operations and improving concurrency.

If the table already has a primary key on some longer column, such as a **VARCHAR**, consider adding a new unsigned **AUTO\_INCREMENT** column and switching the primary key to that, even if that column is not referenced in queries. This design change can produce substantial space savings in the secondary indexes. You can designate the former primary key columns as **UNIQUE NOT NULL** to enforce the same constraints as the **PRIMARY KEY** clause, that is, to prevent duplicate or null values across all those columns.

If you spread related information across multiple tables, typically each table uses the same column for its primary key. For example, a personnel database might have several tables, each with a primary key of employee number. A sales database might have some tables with a primary key of customer number, and other tables with a primary key of order number. Because lookups using the primary key are very fast, you can construct efficient join queries for such tables.

If you leave the **PRIMARY KEY** clause out entirely, MySQL creates an invisible one for you. It is a 6-byte value that might be longer than you need, thus wasting space. Because it is hidden, you cannot refer to it in queries.

##### Application Performance Considerations

The reliability and scalability features of **InnoDB** require more disk storage than equivalent **MyISAM** tables. You might change the column and index definitions slightly, for better space utilization, reduced I/O and memory consumption when processing result sets, and better query optimization plans making efficient use of index lookups.

If you set up a numeric ID column for the primary key, use that value to cross-reference with related values in any other tables, particularly for [join](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_join) queries. For example, rather than accepting a country name as input and doing queries searching for the same name, do one lookup to determine the country ID, then do other queries (or a single join query) to look up relevant information across several tables. Rather than storing a customer or catalog item number as a string of digits, potentially using up several bytes, convert it to a numeric ID for storing and querying. A 4-byte unsigned [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) column can index over 4 billion items (with the US meaning of billion: 1000 million). For the ranges of the different integer types, see [Section 11.1.2, “Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

##### Understanding Files Associated with InnoDB Tables

**InnoDB** files require more care and planning than **MyISAM** files do.

You must not delete the [ibdata files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibdata_file) that represent the **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace).

Methods of moving or copying **InnoDB** tables to a different server are described in [Section 15.6.1.4, “Moving or Copying InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-migration).

#### 15.6.1.6 AUTO\_INCREMENT Handling in InnoDB

**InnoDB** provides a configurable locking mechanism that can significantly improve scalability and performance of SQL statements that add rows to tables with **AUTO\_INCREMENT** columns. To use the **AUTO\_INCREMENT** mechanism with an **InnoDB** table, an **AUTO\_INCREMENT** column must be defined as part of an index such that it is possible to perform the equivalent of an indexed **SELECT MAX(*ai\_col*)** lookup on the table to obtain the maximum column value. Typically, this is achieved by making the **AUTO\_INCREMENT** column the first column of some table index.

This section describes the **AUTO\_INCREMENT** lock modes, usage implications of different **AUTO\_INCREMENT** lock mode settings, and how **InnoDB** initializes the **AUTO\_INCREMENT** counter.

[InnoDB AUTO\_INCREMENT Lock Modes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-lock-modes)

[InnoDB AUTO\_INCREMENT Lock Mode Usage Implications](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-lock-mode-usage-implications)

[InnoDB AUTO\_INCREMENT Counter Initialization](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-initialization)

[Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-notes)

##### InnoDB AUTO\_INCREMENT Lock Modes

This section describes the **AUTO\_INCREMENT** lock modes used to generate auto-increment values, and how each lock mode affects replication. The auto-increment lock mode is configured at startup using the [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) variable.

The following terms are used in describing [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) settings:

“[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements

All statements that generate new rows in a table, including [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), [**REPLACE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data). Includes “simple-inserts”, “bulk-inserts”, and “mixed-mode” inserts.

“Simple inserts”

Statements for which the number of rows to be inserted can be determined in advance (when the statement is initially processed). This includes single-row and multiple-row [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements that do not have a nested subquery, but not [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

“Bulk inserts”

Statements for which the number of rows to be inserted (and the number of required auto-increment values) is not known in advance. This includes [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), [**REPLACE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements, but not plain **INSERT**. **InnoDB** assigns new values for the **AUTO\_INCREMENT** column one at a time as each row is processed.

“Mixed-mode inserts”

These are “simple insert” statements that specify the auto-increment value for some (but not all) of the new rows. An example follows, where **c1** is an **AUTO\_INCREMENT** column of table **t1**:

INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (5,'c'), (NULL,'d');

Another type of “mixed-mode insert” is [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate), which in the worst case is in effect an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) followed by a [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), where the allocated value for the **AUTO\_INCREMENT** column may or may not be used during the update phase.

There are three possible settings for the [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) variable. The settings are 0, 1, or 2, for “traditional”, “consecutive”, or “interleaved” lock mode, respectively. As of MySQL 8.0, interleaved lock mode ([**innodb\_autoinc\_lock\_mode=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode)) is the default setting. Prior to MySQL 8.0, consecutive lock mode is the default ([**innodb\_autoinc\_lock\_mode=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode)).

The default setting of interleaved lock mode in MySQL 8.0 reflects the change from statement-based replication to row based replication as the default replication type. Statement-based replication requires the consecutive auto-increment lock mode to ensure that auto-increment values are assigned in a predictable and repeatable order for a given sequence of SQL statements, whereas row-based replication is not sensitive to the execution order of SQL statements.

**innodb\_autoinc\_lock\_mode = 0** (“traditional” lock mode)

The traditional lock mode provides the same behavior that existed before the [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) variable was introduced. The traditional lock mode option is provided for backward compatibility, performance testing, and working around issues with “mixed-mode inserts”, due to possible differences in semantics.

In this lock mode, all “INSERT-like” statements obtain a special table-level **AUTO-INC** lock for inserts into tables with **AUTO\_INCREMENT** columns. This lock is normally held to the end of the statement (not to the end of the transaction) to ensure that auto-increment values are assigned in a predictable and repeatable order for a given sequence of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements, and to ensure that auto-increment values assigned by any given statement are consecutive.

In the case of statement-based replication, this means that when an SQL statement is replicated on a replica server, the same values are used for the auto-increment column as on the source server. The result of execution of multiple [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements is deterministic, and the replica reproduces the same data as on the source. If auto-increment values generated by multiple [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements were interleaved, the result of two concurrent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements would be nondeterministic, and could not reliably be propagated to a replica server using statement-based replication.

To make this clear, consider an example that uses this table:

CREATE TABLE t1 (

c1 INT(11) NOT NULL AUTO\_INCREMENT,

c2 VARCHAR(10) DEFAULT NULL,

PRIMARY KEY (c1)

) ENGINE=InnoDB;

Suppose that there are two transactions running, each inserting rows into a table with an **AUTO\_INCREMENT** column. One transaction is using an [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statement that inserts 1000 rows, and another is using a simple [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement that inserts one row:

Tx1: INSERT INTO t1 (c2) SELECT 1000 rows from another table ...

Tx2: INSERT INTO t1 (c2) VALUES ('xxx');

**InnoDB** cannot tell in advance how many rows are retrieved from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement in Tx1, and it assigns the auto-increment values one at a time as the statement proceeds. With a table-level lock, held to the end of the statement, only one [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement referring to table **t1** can execute at a time, and the generation of auto-increment numbers by different statements is not interleaved. The auto-increment values generated by the Tx1 [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statement are consecutive, and the (single) auto-increment value used by the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement in Tx2 is either smaller or larger than all those used for Tx1, depending on which statement executes first.

As long as the SQL statements execute in the same order when replayed from the binary log (when using statement-based replication, or in recovery scenarios), the results are the same as they were when Tx1 and Tx2 first ran. Thus, table-level locks held until the end of a statement make [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements using auto-increment safe for use with statement-based replication. However, those table-level locks limit concurrency and scalability when multiple transactions are executing insert statements at the same time.

In the preceding example, if there were no table-level lock, the value of the auto-increment column used for the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) in Tx2 depends on precisely when the statement executes. If the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) of Tx2 executes while the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) of Tx1 is running (rather than before it starts or after it completes), the specific auto-increment values assigned by the two [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements are nondeterministic, and may vary from run to run.

Under the [consecutive](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-lock-mode-consecutive) lock mode, **InnoDB** can avoid using table-level **AUTO-INC** locks for “simple insert” statements where the number of rows is known in advance, and still preserve deterministic execution and safety for statement-based replication.

If you are not using the binary log to replay SQL statements as part of recovery or replication, the [interleaved](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-lock-mode-interleaved) lock mode can be used to eliminate all use of table-level **AUTO-INC** locks for even greater concurrency and performance, at the cost of permitting gaps in auto-increment numbers assigned by a statement and potentially having the numbers assigned by concurrently executing statements interleaved.

**innodb\_autoinc\_lock\_mode = 1** (“consecutive” lock mode)

In this mode, “bulk inserts” use the special **AUTO-INC** table-level lock and hold it until the end of the statement. This applies to all [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), [**REPLACE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements. Only one statement holding the **AUTO-INC** lock can execute at a time. If the source table of the bulk insert operation is different from the target table, the **AUTO-INC** lock on the target table is taken after a shared lock is taken on the first row selected from the source table. If the source and target of the bulk insert operation are the same table, the **AUTO-INC** lock is taken after shared locks are taken on all selected rows.

“Simple inserts” (for which the number of rows to be inserted is known in advance) avoid table-level **AUTO-INC** locks by obtaining the required number of auto-increment values under the control of a mutex (a light-weight lock) that is only held for the duration of the allocation process, not until the statement completes. No table-level **AUTO-INC** lock is used unless an **AUTO-INC** lock is held by another transaction. If another transaction holds an **AUTO-INC** lock, a “simple insert” waits for the **AUTO-INC** lock, as if it were a “bulk insert”.

This lock mode ensures that, in the presence of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements where the number of rows is not known in advance (and where auto-increment numbers are assigned as the statement progresses), all auto-increment values assigned by any “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statement are consecutive, and operations are safe for statement-based replication.

Simply put, this lock mode significantly improves scalability while being safe for use with statement-based replication. Further, as with “traditional” lock mode, auto-increment numbers assigned by any given statement are consecutive. There is no change in semantics compared to “traditional” mode for any statement that uses auto-increment, with one important exception.

The exception is for “mixed-mode inserts”, where the user provides explicit values for an **AUTO\_INCREMENT** column for some, but not all, rows in a multiple-row “simple insert”. For such inserts, **InnoDB** allocates more auto-increment values than the number of rows to be inserted. However, all values automatically assigned are consecutively generated (and thus higher than) the auto-increment value generated by the most recently executed previous statement. “Excess” numbers are lost.

**innodb\_autoinc\_lock\_mode = 2** (“interleaved” lock mode)

In this lock mode, no “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements use the table-level **AUTO-INC** lock, and multiple statements can execute at the same time. This is the fastest and most scalable lock mode, but it is not safe when using statement-based replication or recovery scenarios when SQL statements are replayed from the binary log.

In this lock mode, auto-increment values are guaranteed to be unique and monotonically increasing across all concurrently executing “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements. However, because multiple statements can be generating numbers at the same time (that is, allocation of numbers is interleaved across statements), the values generated for the rows inserted by any given statement may not be consecutive.

If the only statements executing are “simple inserts” where the number of rows to be inserted is known ahead of time, there are no gaps in the numbers generated for a single statement, except for “mixed-mode inserts”. However, when “bulk inserts” are executed, there may be gaps in the auto-increment values assigned by any given statement.

##### InnoDB AUTO\_INCREMENT Lock Mode Usage Implications

Using auto-increment with replication

If you are using statement-based replication, set [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) to 0 or 1 and use the same value on the source and its replicas. Auto-increment values are not ensured to be the same on the replicas as on the source if you use [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) = 2 (“interleaved”) or configurations where the source and replicas do not use the same lock mode.

If you are using row-based or mixed-format replication, all of the auto-increment lock modes are safe, since row-based replication is not sensitive to the order of execution of the SQL statements (and the mixed format uses row-based replication for any statements that are unsafe for statement-based replication).

“Lost” auto-increment values and sequence gaps

In all lock modes (0, 1, and 2), if a transaction that generated auto-increment values rolls back, those auto-increment values are “lost”. Once a value is generated for an auto-increment column, it cannot be rolled back, whether or not the “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statement is completed, and whether or not the containing transaction is rolled back. Such lost values are not reused. Thus, there may be gaps in the values stored in an **AUTO\_INCREMENT** column of a table.

Specifying NULL or 0 for the **AUTO\_INCREMENT** column

In all lock modes (0, 1, and 2), if a user specifies NULL or 0 for the **AUTO\_INCREMENT** column in an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), **InnoDB** treats the row as if the value was not specified and generates a new value for it.

Assigning a negative value to the **AUTO\_INCREMENT** column

In all lock modes (0, 1, and 2), the behavior of the auto-increment mechanism is undefined if you assign a negative value to the **AUTO\_INCREMENT** column.

If the **AUTO\_INCREMENT** value becomes larger than the maximum integer for the specified integer type

In all lock modes (0, 1, and 2), the behavior of the auto-increment mechanism is undefined if the value becomes larger than the maximum integer that can be stored in the specified integer type.

Gaps in auto-increment values for “bulk inserts”

With [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) set to 0 (“traditional”) or 1 (“consecutive”), the auto-increment values generated by any given statement are consecutive, without gaps, because the table-level **AUTO-INC** lock is held until the end of the statement, and only one such statement can execute at a time.

With [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) set to 2 (“interleaved”), there may be gaps in the auto-increment values generated by “bulk inserts,” but only if there are concurrently executing “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements.

For lock modes 1 or 2, gaps may occur between successive statements because for bulk inserts the exact number of auto-increment values required by each statement may not be known and overestimation is possible.

Auto-increment values assigned by “mixed-mode inserts”

Consider a “mixed-mode insert,” where a “simple insert” specifies the auto-increment value for some (but not all) resulting rows. Such a statement behaves differently in lock modes 0, 1, and 2. For example, assume **c1** is an **AUTO\_INCREMENT** column of table **t1**, and that the most recent automatically generated sequence number is 100.

mysql> **CREATE TABLE t1 (**

-> **c1 INT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

-> **c2 CHAR(1)**

-> **) ENGINE = INNODB;**

Now, consider the following “mixed-mode insert” statement:

mysql> **INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (5,'c'), (NULL,'d');**

With [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) set to 0 (“traditional”), the four new rows are:

mysql> **SELECT c1, c2 FROM t1 ORDER BY c2;**

+-----+------+

| c1 | c2 |

+-----+------+

| 1 | a |

| 101 | b |

| 5 | c |

| 102 | d |

+-----+------+

The next available auto-increment value is 103 because the auto-increment values are allocated one at a time, not all at once at the beginning of statement execution. This result is true whether or not there are concurrently executing “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements (of any type).

With [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) set to 1 (“consecutive”), the four new rows are also:

mysql> **SELECT c1, c2 FROM t1 ORDER BY c2;**

+-----+------+

| c1 | c2 |

+-----+------+

| 1 | a |

| 101 | b |

| 5 | c |

| 102 | d |

+-----+------+

However, in this case, the next available auto-increment value is 105, not 103 because four auto-increment values are allocated at the time the statement is processed, but only two are used. This result is true whether or not there are concurrently executing “[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)-like” statements (of any type).

With [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) set to 2 (“interleaved”), the four new rows are:

mysql> **SELECT c1, c2 FROM t1 ORDER BY c2;**

+-----+------+

| c1 | c2 |

+-----+------+

| 1 | a |

| ***x*** | b |

| 5 | c |

| ***y*** | d |

+-----+------+

The values of ***x*** and ***y*** are unique and larger than any previously generated rows. However, the specific values of ***x*** and ***y*** depend on the number of auto-increment values generated by concurrently executing statements.

Finally, consider the following statement, issued when the most-recently generated sequence number is 100:

mysql> **INSERT INTO t1 (c1,c2) VALUES (1,'a'), (NULL,'b'), (101,'c'), (NULL,'d');**

With any [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) setting, this statement generates a duplicate-key error 23000 (**Can't write; duplicate key in table**) because 101 is allocated for the row **(NULL, 'b')** and insertion of the row **(101, 'c')** fails.

Modifying **AUTO\_INCREMENT** column values in the middle of a sequence of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements

In MySQL 5.7 and earlier, modifying an **AUTO\_INCREMENT** column value in the middle of a sequence of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements could lead to “Duplicate entry” errors. For example, if you performed an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operation that changed an **AUTO\_INCREMENT** column value to a value larger than the current maximum auto-increment value, subsequent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations that did not specify an unused auto-increment value could encounter “Duplicate entry” errors. In MySQL 8.0 and later, if you modify an **AUTO\_INCREMENT** column value to a value larger than the current maximum auto-increment value, the new value is persisted, and subsequent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations allocate auto-increment values starting from the new, larger value. This behavior is demonstrated in the following example.

mysql> **CREATE TABLE t1 (**

-> **c1 INT NOT NULL AUTO\_INCREMENT,**

-> **PRIMARY KEY (c1)**

-> **) ENGINE = InnoDB;**

mysql> **INSERT INTO t1 VALUES(0), (0), (3);**

mysql> **SELECT c1 FROM t1;**

+----+

| c1 |

+----+

| 1 |

| 2 |

| 3 |

+----+

mysql> **UPDATE t1 SET c1 = 4 WHERE c1 = 1;**

mysql> **SELECT c1 FROM t1;**

+----+

| c1 |

+----+

| 2 |

| 3 |

| 4 |

+----+

mysql> **INSERT INTO t1 VALUES(0);**

mysql> **SELECT c1 FROM t1;**

+----+

| c1 |

+----+

| 2 |

| 3 |

| 4 |

| 5 |

+----+

##### InnoDB AUTO\_INCREMENT Counter Initialization

This section describes how **InnoDB** initializes **AUTO\_INCREMENT** counters.

If you specify an **AUTO\_INCREMENT** column for an **InnoDB** table, the in-memory table object contains a special counter called the auto-increment counter that is used when assigning new values for the column.

In MySQL 5.7 and earlier, the auto-increment counter is stored in main memory, not on disk. To initialize an auto-increment counter after a server restart, **InnoDB** would execute the equivalent of the following statement on the first insert into a table containing an **AUTO\_INCREMENT** column.

SELECT MAX(ai\_col) FROM ***table\_name*** FOR UPDATE;

In MySQL 8.0, this behavior is changed. The current maximum auto-increment counter value is written to the redo log each time it changes and saved to the data dictionary on each checkpoint. These changes make the current maximum auto-increment counter value persistent across server restarts.

On a server restart following a normal shutdown, **InnoDB** initializes the in-memory auto-increment counter using the current maximum auto-increment value stored in the data dictionary.

On a server restart during crash recovery, **InnoDB** initializes the in-memory auto-increment counter using the current maximum auto-increment value stored in the data dictionary and scans the redo log for auto-increment counter values written since the last checkpoint. If a redo-logged value is greater than the in-memory counter value, the redo-logged value is applied. However, in the case of an unexpected server exit, reuse of a previously allocated auto-increment value cannot be guaranteed. Each time the current maximum auto-increment value is changed due to an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operation, the new value is written to the redo log, but if the unexpected exit occurs before the redo log is flushed to disk, the previously allocated value could be reused when the auto-increment counter is initialized after the server is restarted.

The only circumstance in which **InnoDB** uses the equivalent of a **SELECT MAX(ai\_col) FROM *table\_name* FOR UPDATE** statement to initialize an auto-increment counter is when [importing a table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import) without a .cfg metadata file. Otherwise, the current maximum auto-increment counter value is read from the .cfg metadata file if present. Aside from counter value initialization, the equivalent of a **SELECT MAX(ai\_col) FROM *table\_name*** statement is used to determine the current maximum auto-increment counter value of the table when attempting to set the counter value to one that is smaller than or equal to the persisted counter value using an **ALTER TABLE ... AUTO\_INCREMENT = *N* FOR UPDATE** statement. For example, you might try to set the counter value to a lesser value after deleting some records. In this case, the table must be searched to ensure that the new counter value is not less than or equal to the actual current maximum counter value.

In MySQL 5.7 and earlier, a server restart cancels the effect of the **AUTO\_INCREMENT = N** table option, which may be used in a **CREATE TABLE** or **ALTER TABLE** statement to set an initial counter value or alter the existing counter value, respectively. In MySQL 8.0, a server restart does not cancel the effect of the **AUTO\_INCREMENT = N** table option. If you initialize the auto-increment counter to a specific value, or if you alter the auto-increment counter value to a larger value, the new value is persisted across server restarts.

**Note**

[**ALTER TABLE ... AUTO\_INCREMENT = N**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) can only change the auto-increment counter value to a value larger than the current maximum.

In MySQL 5.7 and earlier, a server restart immediately following a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) operation could result in the reuse of auto-increment values that were previously allocated to the rolled-back transaction, effectively rolling back the current maximum auto-increment value. In MySQL 8.0, the current maximum auto-increment value is persisted, preventing the reuse of previously allocated values.

If a [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) statement examines a table before the auto-increment counter is initialized, **InnoDB** opens the table and initializes the counter value using the current maximum auto-increment value that is stored in the data dictionary. The value is then stored in memory for use by later inserts or updates. Initialization of the counter value uses a normal exclusive-locking read on the table which lasts to the end of the transaction. **InnoDB** follows the same procedure when initializing the auto-increment counter for a newly created table that has a user-specified auto-increment value greater than 0.

After the auto-increment counter is initialized, if you do not explicitly specify an auto-increment value when inserting a row, **InnoDB** implicitly increments the counter and assigns the new value to the column. If you insert a row that explicitly specifies an auto-increment column value, and the value is greater than the current maximum counter value, the counter is set to the specified value.

**InnoDB** uses the in-memory auto-increment counter as long as the server runs. When the server is stopped and restarted, **InnoDB** reinitializes the auto-increment counter, as described earlier.

The [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) variable determines the starting point for the **AUTO\_INCREMENT** column value. The default setting is 1.

The [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) variable controls the interval between successive column values. The default setting is 1.

##### Notes

When an **AUTO\_INCREMENT** integer column runs out of values, a subsequent **INSERT** operation returns a duplicate-key error. This is general MySQL behavior.

### 15.6.2 Indexes

[15.6.2.1 Clustered and Secondary Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types)

[15.6.2.2 The Physical Structure of an InnoDB Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-physical-structure)

[15.6.2.3 Sorted Index Builds](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sorted-index-builds)

[15.6.2.4 InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index)

This section covers topics related to **InnoDB** indexes.

#### 15.6.2.1 Clustered and Secondary Indexes

Each **InnoDB** table has a special index called the clustered index that stores row data. Typically, the clustered index is synonymous with the primary key. To get the best performance from queries, inserts, and other database operations, it is important to understand how **InnoDB** uses the clustered index to optimize the common lookup and DML operations.

When you define a **PRIMARY KEY** on a table, **InnoDB** uses it as the clustered index. A primary key should be defined for each table. If there is no logical unique and non-null column or set of columns to use a the primary key, add an auto-increment column. Auto-increment column values are unique and are added automatically as new rows are inserted.

If you do not define a **PRIMARY KEY** for a table, **InnoDB** uses the first **UNIQUE** index with all key columns defined as **NOT NULL** as the clustered index.

If a table has no **PRIMARY KEY** or suitable **UNIQUE** index, **InnoDB** generates a hidden clustered index named **GEN\_CLUST\_INDEX** on a synthetic column that contains row ID values. The rows are ordered by the row ID that **InnoDB** assigns. The row ID is a 6-byte field that increases monotonically as new rows are inserted. Thus, the rows ordered by the row ID are physically in order of insertion.

##### How the Clustered Index Speeds Up Queries

Accessing a row through the clustered index is fast because the index search leads directly to the page that contains the row data. If a table is large, the clustered index architecture often saves a disk I/O operation when compared to storage organizations that store row data using a different page from the index record.

##### How Secondary Indexes Relate to the Clustered Index

Indexes other than the clustered index are known as secondary indexes. In **InnoDB**, each record in a secondary index contains the primary key columns for the row, as well as the columns specified for the secondary index. **InnoDB** uses this primary key value to search for the row in the clustered index.

If the primary key is long, the secondary indexes use more space, so it is advantageous to have a short primary key.

For guidelines to take advantage of **InnoDB** clustered and secondary indexes, see [Section 8.3, “Optimization and Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimization-indexes).

#### 15.6.2.2 The Physical Structure of an InnoDB Index

With the exception of spatial indexes, **InnoDB** indexes are [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) data structures. Spatial indexes use [R-trees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_r_tree), which are specialized data structures for indexing multi-dimensional data. Index records are stored in the leaf pages of their B-tree or R-tree data structure. The default size of an index page is 16KB. The page size is determined by the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) setting when when the MySQL instance is initialized. See [Section 15.8.1, “InnoDB Startup Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-init-startup-configuration).

When new records are inserted into an **InnoDB** [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index), **InnoDB** tries to leave 1/16 of the page free for future insertions and updates of the index records. If index records are inserted in a sequential order (ascending or descending), the resulting index pages are about 15/16 full. If records are inserted in a random order, the pages are from 1/2 to 15/16 full.

**InnoDB** performs a bulk load when creating or rebuilding B-tree indexes. This method of index creation is known as a sorted index build. The [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) variable defines the percentage of space on each B-tree page that is filled during a sorted index build, with the remaining space reserved for future index growth. Sorted index builds are not supported for spatial indexes. For more information, see [Section 15.6.2.3, “Sorted Index Builds”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sorted-index-builds). An [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) setting of 100 leaves 1/16 of the space in clustered index pages free for future index growth.

If the fill factor of an **InnoDB** index page drops below the **MERGE\_THRESHOLD**, which is 50% by default if not specified, **InnoDB** tries to contract the index tree to free the page. The **MERGE\_THRESHOLD** setting applies to both B-tree and R-tree indexes. For more information, see [Section 15.8.11, “Configuring the Merge Threshold for Index Pages”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold).

#### 15.6.2.3 Sorted Index Builds

**InnoDB** performs a bulk load instead of inserting one index record at a time when creating or rebuilding indexes. This method of index creation is also known as a sorted index build. Sorted index builds are not supported for spatial indexes.

There are three phases to an index build. In the first phase, the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index) is scanned, and index entries are generated and added to the sort buffer. When the [sort buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_sort_buffer) becomes full, entries are sorted and written out to a temporary intermediate file. This process is also known as a “run”. In the second phase, with one or more runs written to the temporary intermediate file, a merge sort is performed on all entries in the file. In the third and final phase, the sorted entries are inserted into the [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree).

Prior to the introduction of sorted index builds, index entries were inserted into the B-tree one record at a time using insert APIs. This method involved opening a B-tree [cursor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cursor) to find the insert position and then inserting entries into a B-tree page using an [optimistic](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_optimistic) insert. If an insert failed due to a page being full, a [pessimistic](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_pessimistic) insert would be performed, which involves opening a B-tree cursor and splitting and merging B-tree nodes as necessary to find space for the entry. The drawbacks of this “top-down” method of building an index are the cost of searching for an insert position and the constant splitting and merging of B-tree nodes.

Sorted index builds use a “bottom-up” approach to building an index. With this approach, a reference to the right-most leaf page is held at all levels of the B-tree. The right-most leaf page at the necessary B-tree depth is allocated and entries are inserted according to their sorted order. Once a leaf page is full, a node pointer is appended to the parent page and a sibling leaf page is allocated for the next insert. This process continues until all entries are inserted, which may result in inserts up to the root level. When a sibling page is allocated, the reference to the previously pinned leaf page is released, and the newly allocated leaf page becomes the right-most leaf page and new default insert location.

##### Reserving B-tree Page Space for Future Index Growth

To set aside space for future index growth, you can use the [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) variable to reserve a percentage of B-tree page space. For example, setting [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) to 80 reserves 20 percent of the space in B-tree pages during a sorted index build. This setting applies to both B-tree leaf and non-leaf pages. It does not apply to external pages used for [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) entries. The amount of space that is reserved may not be exactly as configured, as the [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) value is interpreted as a hint rather than a hard limit.

##### Sorted Index Builds and Full-Text Index Support

Sorted index builds are supported for [fulltext indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fulltext_index). Previously, SQL was used to insert entries into a fulltext index.

##### Sorted Index Builds and Compressed Tables

For [compressed tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression), the previous index creation method appended entries to both compressed and uncompressed pages. When the modification log (representing free space on the compressed page) became full, the compressed page would be recompressed. If compression failed due to a lack of space, the page would be split. With sorted index builds, entries are only appended to uncompressed pages. When an uncompressed page becomes full, it is compressed. Adaptive padding is used to ensure that compression succeeds in most cases, but if compression fails, the page is split and compression is attempted again. This process continues until compression is successful. For more information about compression of B-Tree pages, see [Section 15.9.1.5, “How Compression Works for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals).

##### Sorted Index Builds and Redo Logging

[Redo logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) is disabled during a sorted index build. Instead, there is a [checkpoint](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint) to ensure that the index build can withstand an unexpected exit or failure. The checkpoint forces a write of all dirty pages to disk. During a sorted index build, the [page cleaner](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_cleaner) thread is signaled periodically to flush [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) to ensure that the checkpoint operation can be processed quickly. Normally, the page cleaner thread flushes dirty pages when the number of clean pages falls below a set threshold. For sorted index builds, dirty pages are flushed promptly to reduce checkpoint overhead and to parallelize I/O and CPU activity.

##### Sorted Index Builds and Optimizer Statistics

Sorted index builds may result in [optimizer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_optimizer) statistics that differ from those generated by the previous method of index creation. The difference in statistics, which is not expected to affect workload performance, is due to the different algorithm used to populate the index.

#### 15.6.2.4 InnoDB Full-Text Indexes

Full-text indexes are created on text-based columns ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns) to speed up queries and DML operations on data contained within those columns.

A full-text index is defined as part of a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement or added to an existing table using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

Full-text search is performed using [**MATCH() ... AGAINST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_match) syntax. For usage information, see [Section 12.10, “Full-Text Search Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search).

**InnoDB** full-text indexes are described under the following topics in this section:

[InnoDB Full-Text Index Design](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-design)

[InnoDB Full-Text Index Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-tables)

[InnoDB Full-Text Index Cache](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-cache)

[InnoDB Full-Text Index DOC\_ID and FTS\_DOC\_ID Column](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-docid)

[InnoDB Full-Text Index Deletion Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-deletion)

[InnoDB Full-Text Index Transaction Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-transaction)

[Monitoring InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-monitoring)

##### InnoDB Full-Text Index Design

**InnoDB** full-text indexes have an inverted index design. Inverted indexes store a list of words, and for each word, a list of documents that the word appears in. To support proximity search, position information for each word is also stored, as a byte offset.

##### InnoDB Full-Text Index Tables

When an **InnoDB** full-text index is created, a set of index tables is created, as shown in the following example:

mysql> **CREATE TABLE opening\_lines (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**opening\_line TEXT(500),**

**author VARCHAR(200),**

**title VARCHAR(200),**

**FULLTEXT idx (opening\_line)**

**) ENGINE=InnoDB;**

mysql> **SELECT table\_id, name, space from INFORMATION\_SCHEMA.INNODB\_TABLES**

**WHERE name LIKE 'test/%';**

+----------+----------------------------------------------------+-------+

| table\_id | name | space |

+----------+----------------------------------------------------+-------+

| 333 | test/fts\_0000000000000147\_00000000000001c9\_index\_1 | 289 |

| 334 | test/fts\_0000000000000147\_00000000000001c9\_index\_2 | 290 |

| 335 | test/fts\_0000000000000147\_00000000000001c9\_index\_3 | 291 |

| 336 | test/fts\_0000000000000147\_00000000000001c9\_index\_4 | 292 |

| 337 | test/fts\_0000000000000147\_00000000000001c9\_index\_5 | 293 |

| 338 | test/fts\_0000000000000147\_00000000000001c9\_index\_6 | 294 |

| 330 | test/fts\_0000000000000147\_being\_deleted | 286 |

| 331 | test/fts\_0000000000000147\_being\_deleted\_cache | 287 |

| 332 | test/fts\_0000000000000147\_config | 288 |

| 328 | test/fts\_0000000000000147\_deleted | 284 |

| 329 | test/fts\_0000000000000147\_deleted\_cache | 285 |

| 327 | test/opening\_lines | 283 |

+----------+----------------------------------------------------+-------+

The first six index tables comprise the inverted index and are referred to as auxiliary index tables. When incoming documents are tokenized, the individual words (also referred to as “tokens”) are inserted into the index tables along with position information and an associated **DOC\_ID**. The words are fully sorted and partitioned among the six index tables based on the character set sort weight of the word's first character.

The inverted index is partitioned into six auxiliary index tables to support parallel index creation. By default, two threads tokenize, sort, and insert words and associated data into the index tables. The number of threads that perform this work is configurable using the [**innodb\_ft\_sort\_pll\_degree**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_sort_pll_degree) variable. Consider increasing the number of threads when creating full-text indexes on large tables.

Auxiliary index table names are prefixed with **fts\_** and postfixed with **index\_*#***. Each auxiliary index table is associated with the indexed table by a hex value in the auxiliary index table name that matches the **table\_id** of the indexed table. For example, the **table\_id** of the **test/opening\_lines** table is **327**, for which the hex value is 0x147. As shown in the preceding example, the “147” hex value appears in the names of auxiliary index tables that are associated with the **test/opening\_lines** table.

A hex value representing the **index\_id** of the full-text index also appears in auxiliary index table names. For example, in the auxiliary table name **test/fts\_0000000000000147\_00000000000001c9\_index\_1**, the hex value **1c9** has a decimal value of 457. The index defined on the **opening\_lines** table (**idx**) can be identified by querying the [**INFORMATION\_SCHEMA.INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) table for this value (457).

mysql> **SELECT index\_id, name, table\_id, space from INFORMATION\_SCHEMA.INNODB\_INDEXES**

**WHERE index\_id=457;**

+----------+------+----------+-------+

| index\_id | name | table\_id | space |

+----------+------+----------+-------+

| 457 | idx | 327 | 283 |

+----------+------+----------+-------+

Index tables are stored in their own tablespace if the primary table is created in a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace. Otherwise, index tables are stored in the tablespace where the indexed table resides.

The other index tables shown in the preceding example are referred to as common index tables and are used for deletion handling and storing the internal state of full-text indexes. Unlike the inverted index tables, which are created for each full-text index, this set of tables is common to all full-text indexes created on a particular table.

Common index tables are retained even if full-text indexes are dropped. When a full-text index is dropped, the **FTS\_DOC\_ID** column that was created for the index is retained, as removing the **FTS\_DOC\_ID** column would require rebuilding the previously indexed table. Common index tables are required to manage the **FTS\_DOC\_ID** column.

**fts\_\*\_deleted** and **fts\_\*\_deleted\_cache**

Contain the document IDs (DOC\_ID) for documents that are deleted but whose data is not yet removed from the full-text index. The **fts\_\*\_deleted\_cache** is the in-memory version of the **fts\_\*\_deleted** table.

**fts\_\*\_being\_deleted** and **fts\_\*\_being\_deleted\_cache**

Contain the document IDs (DOC\_ID) for documents that are deleted and whose data is currently in the process of being removed from the full-text index. The **fts\_\*\_being\_deleted\_cache** table is the in-memory version of the **fts\_\*\_being\_deleted** table.

**fts\_\*\_config**

Stores information about the internal state of the full-text index. Most importantly, it stores the **FTS\_SYNCED\_DOC\_ID**, which identifies documents that have been parsed and flushed to disk. In case of crash recovery, **FTS\_SYNCED\_DOC\_ID** values are used to identify documents that have not been flushed to disk so that the documents can be re-parsed and added back to the full-text index cache. To view the data in this table, query the [**INFORMATION\_SCHEMA.INNODB\_FT\_CONFIG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-config-table) table.

##### InnoDB Full-Text Index Cache

When a document is inserted, it is tokenized, and the individual words and associated data are inserted into the full-text index. This process, even for small documents, can result in numerous small insertions into the auxiliary index tables, making concurrent access to these tables a point of contention. To avoid this problem, **InnoDB** uses a full-text index cache to temporarily cache index table insertions for recently inserted rows. This in-memory cache structure holds insertions until the cache is full and then batch flushes them to disk (to the auxiliary index tables). You can query the [**INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table) table to view tokenized data for recently inserted rows.

The caching and batch flushing behavior avoids frequent updates to auxiliary index tables, which could result in concurrent access issues during busy insert and update times. The batching technique also avoids multiple insertions for the same word, and minimizes duplicate entries. Instead of flushing each word individually, insertions for the same word are merged and flushed to disk as a single entry, improving insertion efficiency while keeping auxiliary index tables as small as possible.

The [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) variable is used to configure the full-text index cache size (on a per-table basis), which affects how often the full-text index cache is flushed. You can also define a global full-text index cache size limit for all tables in a given instance using the [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size) variable.

The full-text index cache stores the same information as auxiliary index tables. However, the full-text index cache only caches tokenized data for recently inserted rows. The data that is already flushed to disk (to the auxiliary index tables) is not brought back into the full-text index cache when queried. The data in auxiliary index tables is queried directly, and results from the auxiliary index tables are merged with results from the full-text index cache before being returned.

##### InnoDB Full-Text Index DOC\_ID and FTS\_DOC\_ID Column

**InnoDB** uses a unique document identifier referred to as the **DOC\_ID** to map words in the full-text index to document records where the word appears. The mapping requires an **FTS\_DOC\_ID** column on the indexed table. If an **FTS\_DOC\_ID** column is not defined, **InnoDB** automatically adds a hidden **FTS\_DOC\_ID** column when the full-text index is created. The following example demonstrates this behavior.

The following table definition does not include an **FTS\_DOC\_ID** column:

mysql> **CREATE TABLE opening\_lines (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**opening\_line TEXT(500),**

**author VARCHAR(200),**

**title VARCHAR(200)**

**) ENGINE=InnoDB;**

When you create a full-text index on the table using **CREATE FULLTEXT INDEX** syntax, a warning is returned which reports that **InnoDB** is rebuilding the table to add the **FTS\_DOC\_ID** column.

mysql> **CREATE FULLTEXT INDEX idx ON opening\_lines(opening\_line);**

Query OK, 0 rows affected, 1 warning (0.19 sec)

Records: 0 Duplicates: 0 Warnings: 1

mysql> **SHOW WARNINGS;**

+---------+------+--------------------------------------------------+

| Level | Code | Message |

+---------+------+--------------------------------------------------+

| Warning | 124 | InnoDB rebuilding table to add column FTS\_DOC\_ID |

+---------+------+--------------------------------------------------+

The same warning is returned when using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to add a full-text index to a table that does not have an **FTS\_DOC\_ID** column. If you create a full-text index at [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) time and do not specify an **FTS\_DOC\_ID** column, **InnoDB** adds a hidden **FTS\_DOC\_ID** column, without warning.

Defining an **FTS\_DOC\_ID** column at [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) time is less expensive than creating a full-text index on a table that is already loaded with data. If an **FTS\_DOC\_ID** column is defined on a table prior to loading data, the table and its indexes do not have to be rebuilt to add the new column. If you are not concerned with **CREATE FULLTEXT INDEX** performance, leave out the **FTS\_DOC\_ID** column to have **InnoDB** create it for you. **InnoDB** creates a hidden **FTS\_DOC\_ID** column along with a unique index (**FTS\_DOC\_ID\_INDEX**) on the **FTS\_DOC\_ID** column. If you want to create your own **FTS\_DOC\_ID** column, the column must be defined as **BIGINT UNSIGNED NOT NULL** and named **FTS\_DOC\_ID** (all uppercase), as in the following example:

**Note**

The **FTS\_DOC\_ID** column does not need to be defined as an **AUTO\_INCREMENT** column, but doing so could make loading data easier.

mysql> **CREATE TABLE opening\_lines (**

**FTS\_DOC\_ID BIGINT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**opening\_line TEXT(500),**

**author VARCHAR(200),**

**title VARCHAR(200)**

**) ENGINE=InnoDB;**

If you choose to define the **FTS\_DOC\_ID** column yourself, you are responsible for managing the column to avoid empty or duplicate values. **FTS\_DOC\_ID** values cannot be reused, which means **FTS\_DOC\_ID** values must be ever increasing.

Optionally, you can create the required unique **FTS\_DOC\_ID\_INDEX** (all uppercase) on the **FTS\_DOC\_ID** column.

mysql> **CREATE UNIQUE INDEX FTS\_DOC\_ID\_INDEX on opening\_lines(FTS\_DOC\_ID);**

If you do not create the **FTS\_DOC\_ID\_INDEX**, **InnoDB** creates it automatically.

**Note**

**FTS\_DOC\_ID\_INDEX** cannot be defined as a descending index because the **InnoDB** SQL parser does not use descending indexes.

The permitted gap between the largest used **FTS\_DOC\_ID** value and new **FTS\_DOC\_ID** value is 65535.

To avoid rebuilding the table, the **FTS\_DOC\_ID** column is retained when dropping a full-text index.

##### InnoDB Full-Text Index Deletion Handling

Deleting a record that has a full-text index column could result in numerous small deletions in the auxiliary index tables, making concurrent access to these tables a point of contention. To avoid this problem, the **DOC\_ID** of a deleted document is logged in a special **FTS\_\*\_DELETED** table whenever a record is deleted from an indexed table, and the indexed record remains in the full-text index. Before returning query results, information in the **FTS\_\*\_DELETED** table is used to filter out deleted **DOC\_ID**s. The benefit of this design is that deletions are fast and inexpensive. The drawback is that the size of the index is not immediately reduced after deleting records. To remove full-text index entries for deleted records, run **OPTIMIZE TABLE** on the indexed table with [**innodb\_optimize\_fulltext\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) to rebuild the full-text index. For more information, see [Optimizing InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-optimize).

##### InnoDB Full-Text Index Transaction Handling

**InnoDB** full-text indexes have special transaction handling characteristics due its caching and batch processing behavior. Specifically, updates and insertions on a full-text index are processed at transaction commit time, which means that a full-text search can only see committed data. The following example demonstrates this behavior. The full-text search only returns a result after the inserted lines are committed.

mysql> **CREATE TABLE opening\_lines (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**opening\_line TEXT(500),**

**author VARCHAR(200),**

**title VARCHAR(200),**

**FULLTEXT idx (opening\_line)**

**) ENGINE=InnoDB;**

mysql> **BEGIN;**

mysql> **INSERT INTO opening\_lines(opening\_line,author,title) VALUES**

**('Call me Ishmael.','Herman Melville','Moby-Dick'),**

**('A screaming comes across the sky.','Thomas Pynchon','Gravity\'s Rainbow'),**

**('I am an invisible man.','Ralph Ellison','Invisible Man'),**

**('Where now? Who now? When now?','Samuel Beckett','The Unnamable'),**

**('It was love at first sight.','Joseph Heller','Catch-22'),**

**('All this happened, more or less.','Kurt Vonnegut','Slaughterhouse-Five'),**

**('Mrs. Dalloway said she would buy the flowers herself.','Virginia Woolf','Mrs. Dalloway'),**

**('It was a pleasure to burn.','Ray Bradbury','Fahrenheit 451');**

mysql> **SELECT COUNT(\*) FROM opening\_lines WHERE MATCH(opening\_line) AGAINST('Ishmael');**

+----------+

| COUNT(\*) |

+----------+

| 0 |

+----------+

mysql> **COMMIT;**

mysql> **SELECT COUNT(\*) FROM opening\_lines WHERE MATCH(opening\_line) AGAINST('Ishmael');**

+----------+

| COUNT(\*) |

+----------+

| 1 |

+----------+

##### Monitoring InnoDB Full-Text Indexes

You can monitor and examine the special text-processing aspects of **InnoDB** full-text indexes by querying the following **INFORMATION\_SCHEMA** tables:

[**INNODB\_FT\_CONFIG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-config-table)

[**INNODB\_FT\_INDEX\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-table-table)

[**INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table)

[**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table)

[**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table)

[**INNODB\_FT\_BEING\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-being-deleted-table)

You can also view basic information for full-text indexes and tables by querying [**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) and [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table).

For more information, see [Section 15.15.4, “InnoDB INFORMATION\_SCHEMA FULLTEXT Index Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-fulltext_index-tables).

### 15.6.3 Tablespaces

[15.6.3.1 The System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-system-tablespace)

[15.6.3.2 File-Per-Table Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces)

[15.6.3.3 General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces)

[15.6.3.4 Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces)

[15.6.3.5 Temporary Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-temporary-tablespace)

[15.6.3.6 Moving Tablespace Files While the Server is Offline](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-moving-data-files-offline)

[15.6.3.7 Disabling Tablespace Path Validation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disabling-tablespace-path-validation)

[15.6.3.8 Optimizing Tablespace Space Allocation on Linux](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-optimize-tablespace-page-allocation)

[15.6.3.9 Tablespace AUTOEXTEND\_SIZE Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size)

This section covers topics related to **InnoDB** tablespaces.

#### 15.6.3.1 The System Tablespace

The system tablespace is the storage area for the change buffer. It may also contain table and index data if tables are created in the system tablespace rather than file-per-table or general tablespaces. In previous MySQL versions, the system tablespace contained the **InnoDB** data dictionary. In MySQL 8.0, **InnoDB** stores metadata in the MySQL data dictionary. See [Chapter 14, *MySQL Data Dictionary*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-dictionary.html). In previous MySQL releases, the system tablespace also contained the doublewrite buffer storage area. This storage area resides in separate doublewrite files as of MySQL 8.0.20. See [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

The system tablespace can have one or more data files. By default, a single system tablespace data file, named ibdata1, is created in the data directory. The size and number of system tablespace data files is defined by the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) startup option. For configuration information, see [System Tablespace Data File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-data-file-configuration).

Additional information about the system tablespace is provided under the following topics in the section:

[Resizing the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-resize-system-tablespace)

[Using Raw Disk Partitions for the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-raw-devices)

##### Resizing the System Tablespace

This section describes how to increase or decrease the size of the system tablespace.

###### Increasing the Size of the System Tablespace

The easiest way to increase the size of the system tablespace is to configure it to be auto-extending. To do so, specify the **autoextend** attribute for the last data file in the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting, and restart the server. For example:

innodb\_data\_file\_path=ibdata1:10M:autoextend

When the **autoextend** attribute is specified, the data file automatically increases in size by 8MB increments as space is required. The [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) variable controls the increment size.

You can also increase system tablespace size by adding another data file. To do so:

Stop the MySQL server.

If the last data file in the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting is defined with the **autoextend** attribute, remove it, and modify the size attribute to reflect the current data file size. To determine the appropriate data file size to specify, check your file system for the file size, and round that value down to the closest MB value, where a MB is equal to 1024 x 1024 bytes.

Append a new data file to the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting, optionally specifying the **autoextend** attribute. The **autoextend** attribute can be specified only for the last data file in the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting.

Start the MySQL server.

For example, this tablespace has one auto-extending data file:

innodb\_data\_home\_dir =

innodb\_data\_file\_path = /ibdata/ibdata1:10M:autoextend

Suppose that the data file has grown to 988MB over time. This is the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting after modifying the size attribute to reflect the current data file size, and after specifying a new 50MB auto-extending data file:

innodb\_data\_home\_dir =

innodb\_data\_file\_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend

When adding a new data file, do not specify an existing file name. **InnoDB** creates and initializes the new data file when you start the server.

**Note**

You cannot increase the size of an existing system tablespace data file by changing its size attribute. For example, changing the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting from **ibdata1:10M:autoextend** to **ibdata1:12M:autoextend** produces the following error when starting the server:

[ERROR] [MY-012263] [InnoDB] The Auto-extending innodb\_system

data file './ibdata1' is of a different size 640 pages (rounded down to MB) than

specified in the .cnf file: initial 768 pages, max 0 (relevant if non-zero) pages!

The error indicates that the existing data file size (expressed in **InnoDB** pages) is different from the data file size specified in the configuration file. If you encounter this error, restore the previous [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting, and refer to the system tablespace resizing instructions.

###### Decreasing the Size of the InnoDB System Tablespace

Decreasing the size of an existing system tablespace is not supported. The only option to achieve a smaller system tablespace is to restore your data from a backup to a new MySQL instance created with the desired system tablespace size configuration.

For information about creating backups, see [Section 15.18.1, “InnoDB Backup”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-backup).

For information about configuring data files for a new system tablespace. See [System Tablespace Data File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-data-file-configuration).

To avoid a large system tablespace, consider using file-per-table tablespaces or general tablespaces for your data. File-per-table tablespaces are the default tablespace type and are used implicitly when creating an **InnoDB** table. Unlike the system tablespace, file-per-table tablespaces return disk space to the operating system when they are truncated or dropped. For more information, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces). General tablespaces are multi-table tablespaces that can also be used as an alternative to the system tablespace. See [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

##### Using Raw Disk Partitions for the System Tablespace

Raw disk partitions can be used as system tablespace data files. This technique enables nonbuffered I/O on Windows and some Linux and Unix systems without file system overhead. Perform tests with and without raw partitions to verify whether they improve performance on your system.

When using a raw disk partition, ensure that the user ID that runs the MySQL server has read and write privileges for that partition. For example, if running the server as the **mysql** user, the partition must be readable and writeable by **mysql**. If running the server with the [--memlock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_memlock) option, the server must be run as **root**, so the partition must be readable and writeable by **root**.

The procedures described below involve option file modification. For additional information, see [Section 4.2.2.2, “Using Option Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files).

###### Allocating a Raw Disk Partition on Linux and Unix Systems

When creating a new data file, specify the keyword **newraw** immediately after the data file size for the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) option. The partition must be at least as large as the size that you specify. Note that 1MB in **InnoDB** is 1024 × 1024 bytes, whereas 1MB in disk specifications usually means 1,000,000 bytes.

[mysqld]

innodb\_data\_home\_dir=

innodb\_data\_file\_path=/dev/hdd1:3Gnewraw;/dev/hdd2:2Gnewraw

Restart the server. **InnoDB** notices the **newraw** keyword and initializes the new partition. However, do not create or change any **InnoDB** tables yet. Otherwise, when you next restart the server, **InnoDB** reinitializes the partition and your changes are lost. (As a safety measure **InnoDB** prevents users from modifying data when any partition with **newraw** is specified.)

After **InnoDB** has initialized the new partition, stop the server, change **newraw** in the data file specification to **raw**:

[mysqld]

innodb\_data\_home\_dir=

innodb\_data\_file\_path=/dev/hdd1:3Graw;/dev/hdd2:2Graw

Restart the server. **InnoDB** now permits changes to be made.

###### Allocating a Raw Disk Partition on Windows

On Windows systems, the same steps and accompanying guidelines described for Linux and Unix systems apply except that the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting differs slightly on Windows.

When creating a new data file, specify the keyword **newraw** immediately after the data file size for the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) option:

[mysqld]

innodb\_data\_home\_dir=

innodb\_data\_file\_path=//./D::10Gnewraw

The //./ corresponds to the Windows syntax of \\.\ for accessing physical drives. In the example above, **D:** is the drive letter of the partition.

Restart the server. **InnoDB** notices the **newraw** keyword and initializes the new partition.

After **InnoDB** has initialized the new partition, stop the server, change **newraw** in the data file specification to **raw**:

[mysqld]

innodb\_data\_home\_dir=

innodb\_data\_file\_path=//./D::10Graw

Restart the server. **InnoDB** now permits changes to be made.

#### 15.6.3.2 File-Per-Table Tablespaces

A file-per-table tablespace contains data and indexes for a single **InnoDB** table, and is stored on the file system in a single data file.

File-per-table tablespace characteristics are described under the following topics in this section:

[File-Per-Table Tablespace Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-configuration)

[File-Per-Table Tablespace Data Files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-data-files)

[File-Per-Table Tablespace Advantages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-advantages)

[File-Per-Table Tablespace Disadvantages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-disadvantages)

##### File-Per-Table Tablespace Configuration

**InnoDB** creates tables in file-per-table tablespaces by default. This behavior is controlled by the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable. Disabling [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) causes **InnoDB** to create tables in the system tablespace.

An [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting can be specified in an option file or configured at runtime using a [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. Changing the setting at runtime requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

Option file:

[mysqld]

innodb\_file\_per\_table=ON

Using [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) at runtime:

mysql> SET GLOBAL innodb\_file\_per\_table=ON;

##### File-Per-Table Tablespace Data Files

A file-per-table tablespace is created in an .idb data file in a schema directory under the MySQL data directory. The .ibd file is named for the table (***table\_name***.ibd). For example, the data file for table **test.t1** is created in the test directory under the MySQL data directory:

mysql> USE test;

mysql> CREATE TABLE t1 (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100)

) ENGINE = InnoDB;

shell> cd /***path***/***to***/***mysql***/data/test

shell> ls

t1.ibd

You can use the **DATA DIRECTORY** clause of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement to implicitly create a file-per-table tablespace data file outside of the data directory. For more information, see [Section 15.6.1.2, “Creating Tables Externally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external).

##### File-Per-Table Tablespace Advantages

File-per-table tablespaces have the following advantages over shared tablespaces such as the system tablespace or general tablespaces.

Disk space is returned to the operating system after truncating or dropping a table created in a file-per-table tablespace. Truncating or dropping a table stored in a shared tablespace creates free space within the shared tablespace data file, which can only be used for **InnoDB** data. In other words, a shared tablespace data file does not shrink in size after a table is truncated or dropped.

A table-copying [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation on a table that resides in a shared tablespace can increase the amount of disk space occupied by the tablespace. Such operations may require as much additional space as the data in the table plus indexes. This space is not released back to the operating system as it is for file-per-table tablespaces.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) performance is better when executed on tables that reside in file-per-table tablespaces.

File-per-table tablespace data files can be created on separate storage devices for I/O optimization, space management, or backup purposes. See [Section 15.6.1.2, “Creating Tables Externally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external).

You can import a table that resides in file-per-table tablespace from another MySQL instance. See [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

Tables created in file-per-table tablespaces support features associated with **DYNAMIC** and **COMPRESSED** row formats, which are not supported by the system tablespace. See [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

Tables stored in individual tablespace data files can save time and improve chances for a successful recovery when data corruption occurs, when backups or binary logs are unavailable, or when the MySQL server instance cannot be restarted.

Tables created in file-per-table tablespaces cab be backed up or restored quickly using MySQL Enterprise Backup, without interrupting the use of other **InnoDB** tables. This is beneficial for tables on varying backup schedules or that require backup less frequently. See [Making a Partial Backup](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/partial.html) for details.

File-per-table tablespaces permit monitoring table size on the file system by monitoring the size of the tablespace data file.

Common Linux file systems do not permit concurrent writes to a single file such as a shared tablespace data file when [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) is set to **O\_DIRECT**. As a result, there are possible performance improvements when using file-per-table tablespaces in conjunction with this setting.

Tables in a shared tablespace are limited in size by the 64TB tablespace size limit. By comparison, each file-per-table tablespace has a 64TB size limit, which provides plenty of room for individual tables to grow in size.

##### File-Per-Table Tablespace Disadvantages

File-per-table tablespaces have the following disadvantages compared to shared tablespaces such as the system tablespace or general tablespaces.

With file-per-table tablespaces, each table may have unused space that can only be utilized by rows of the same table, which can lead to wasted space if not properly managed.

**fsync** operations are performed on multiple file-per-table data files instead of a single shared tablespace data file. Because **fsync** operations are per file, write operations for multiple tables cannot be combined, which can result in a higher total number of **fsync** operations.

[**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) must keep an open file handle for each file-per-table tablespace, which may impact performance if you have numerous tables in file-per-table tablespaces.

More file descriptors are required when each table has its own data file.

There is potential for more fragmentation, which can impede [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) and table scan performance. However, if fragmentation is managed, file-per-table tablespaces can improve performance for these operations.

The buffer pool is scanned when dropping a table that resides in a file-per-table tablespace, which can take several seconds for large buffer pools. The scan is performed with a broad internal lock, which may delay other operations.

The [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) variable, which defines the increment size for extending the size of an auto-extending shared tablespace file when it becomes full, does not apply to file-per-table tablespace files, which are auto-extending regardless of the [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) setting. Initial file-per-table tablespace extensions are by small amounts, after which extensions occur in increments of 4MB.

#### 15.6.3.3 General Tablespaces

A general tablespace is a shared **InnoDB** tablespace that is created using [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax. General tablespace capabilities and features are described under the following topics in this section:

[General Tablespace Capabilities](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-capabilities)

[Creating a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-creating)

[Adding Tables to a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-adding-tables)

[General Tablespace Row Format Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-row-format-support)

[Moving Tables Between Tablespaces Using ALTER TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-moving-non-partitioned-tables)

[Renaming a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-renaming)

[Dropping a General Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-dropping)

[General Tablespace Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-limitations)

##### General Tablespace Capabilities

General tablespaces provide the following capabilities:

Similar to the system tablespace, general tablespaces are shared tablespaces capable of storing data for multiple tables.

General tablespaces have a potential memory advantage over [file-per-table tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces). The server keeps tablespace metadata in memory for the lifetime of a tablespace. Multiple tables in fewer general tablespaces consume less memory for tablespace metadata than the same number of tables in separate file-per-table tablespaces.

General tablespace data files can be placed in a directory relative to or independent of the MySQL data directory, which provides you with many of the data file and storage management capabilities of [file-per-table tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces). As with file-per-table tablespaces, the ability to place data files outside of the MySQL data directory allows you to manage performance of critical tables separately, setup RAID or DRBD for specific tables, or bind tables to particular disks, for example.

General tablespaces support all table row formats and associated features.

The **TABLESPACE** option can be used with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) to create tables in a general tablespaces, file-per-table tablespace, or in the system tablespace.

The **TABLESPACE** option can be used with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to move tables between general tablespaces, file-per-table tablespaces, and the system tablespace.

##### Creating a General Tablespace

General tablespaces are created using [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax.

CREATE TABLESPACE ***tablespace\_name***

[ADD DATAFILE '***file\_name***']

[FILE\_BLOCK\_SIZE = ***value***]

[ENGINE [=] ***engine\_name***]

A general tablespace can be created in the data directory or outside of it. To avoid conflicts with implicitly created file-per-table tablespaces, creating a general tablespace in a subdirectory under the data directory is not supported. When creating a general tablespace outside of the data directory, the directory must exist and must be known to **InnoDB** prior to creating the tablespace. To make an unknown directory known to **InnoDB**, add the directory to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value. [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) is a read-only startup option. Configuring it requires restarting the server.

Examples:

Creating a general tablespace in the data directory:

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' Engine=InnoDB;**

or

mysql> **CREATE TABLESPACE `ts1` Engine=InnoDB;**

The **ADD DATAFILE** clause is optional as of MySQL 8.0.14 and required before that. If the **ADD DATAFILE** clause is not specified when creating a tablespace, a tablespace data file with a unique file name is created implicitly. The unique file name is a 128 bit UUID formatted into five groups of hexadecimal numbers separated by dashes (***aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee***). General tablespace data files include an .ibd file extension. In a replication environment, the data file name created on the source is not the same as the data file name created on the replica.

Creating a general tablespace in a directory outside of the data directory:

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE '/my/tablespace/directory/ts1.ibd' Engine=InnoDB;**

You can specify a path that is relative to the data directory as long as the tablespace directory is not under the data directory. In this example, the my\_tablespace directory is at the same level as the data directory:

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE '../my\_tablespace/ts1.ibd' Engine=InnoDB;**

**Note**

The **ENGINE = InnoDB** clause must be defined as part of the [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) statement, or **InnoDB** must be defined as the default storage engine ([**default\_storage\_engine=InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine)).

##### Adding Tables to a General Tablespace

After creating a general tablespace, [**CREATE TABLE *tbl\_name* ... TABLESPACE [=] *tablespace\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE *tbl\_name* TABLESPACE [=] *tablespace\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements can be used to add tables to the tablespace, as shown in the following examples:

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table):

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1;**

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

mysql> **ALTER TABLE t2 TABLESPACE ts1;**

**Note**

Support for adding table partitions to shared tablespaces was deprecated in MySQL 5.7.24 and removed in MySQL 8.0.13. Shared tablespaces include the **InnoDB** system tablespace and general tablespaces.

For detailed syntax information, see [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

##### General Tablespace Row Format Support

General tablespaces support all table row formats (**REDUNDANT**, **COMPACT**, **DYNAMIC**, **COMPRESSED**) with the caveat that compressed and uncompressed tables cannot coexist in the same general tablespace due to different physical page sizes.

For a general tablespace to contain compressed tables (**ROW\_FORMAT=COMPRESSED**), the **FILE\_BLOCK\_SIZE** option must be specified, and the **FILE\_BLOCK\_SIZE** value must be a valid compressed page size in relation to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. Also, the physical page size of the compressed table (**KEY\_BLOCK\_SIZE**) must be equal to **FILE\_BLOCK\_SIZE/1024**. For example, if [**innodb\_page\_size=16KB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) and **FILE\_BLOCK\_SIZE=8K**, the **KEY\_BLOCK\_SIZE** of the table must be 8.

The following table shows permitted [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), **FILE\_BLOCK\_SIZE**, and **KEY\_BLOCK\_SIZE** combinations. **FILE\_BLOCK\_SIZE** values may also be specified in bytes. To determine a valid **KEY\_BLOCK\_SIZE** value for a given **FILE\_BLOCK\_SIZE**, divide the **FILE\_BLOCK\_SIZE** value by 1024. Table compression is not support for 32K and 64K **InnoDB** page sizes. For more information about **KEY\_BLOCK\_SIZE**, see [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), and [Section 15.9.1.2, “Creating Compressed Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-usage).

**Table 15.3 Permitted Page Size, FILE\_BLOCK\_SIZE, and KEY\_BLOCK\_SIZE Combinations for Compressed Tables**

| **InnoDB Page Size (innodb\_page\_size)** | **Permitted FILE\_BLOCK\_SIZE Value** | **Permitted KEY\_BLOCK\_SIZE Value** |
| --- | --- | --- |
| **64KB** | 64K (65536) | Compression is not supported |
| **32KB** | 32K (32768) | Compression is not supported |
| **16KB** | 16K (16384) | None. If [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is equal to **FILE\_BLOCK\_SIZE**, the tablespace cannot contain a compressed table. |
| **16KB** | 8K (8192) | 8 |
| **16KB** | 4K (4096) | 4 |
| **16KB** | 2K (2048) | 2 |
| **16KB** | 1K (1024) | 1 |
| **8KB** | 8K (8192) | None. If [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is equal to **FILE\_BLOCK\_SIZE**, the tablespace cannot contain a compressed table. |
| **8KB** | 4K (4096) | 4 |
| **8KB** | 2K (2048) | 2 |
| **8KB** | 1K (1024) | 1 |
| **4KB** | 4K (4096) | None. If [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is equal to **FILE\_BLOCK\_SIZE**, the tablespace cannot contain a compressed table. |
| **4KB** | 2K (2048) | 2 |
| **4KB** | 1K (1024) | 1 |

This example demonstrates creating a general tablespace and adding a compressed table. The example assumes a default [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) of 16KB. The **FILE\_BLOCK\_SIZE** of 8192 requires that the compressed table have a **KEY\_BLOCK\_SIZE** of 8.

mysql> **CREATE TABLESPACE `ts2` ADD DATAFILE 'ts2.ibd' FILE\_BLOCK\_SIZE = 8192 Engine=InnoDB;**

mysql> **CREATE TABLE t4 (c1 INT PRIMARY KEY) TABLESPACE ts2 ROW\_FORMAT=COMPRESSED KEY\_BLOCK\_SIZE=8;**

If you do not specify **FILE\_BLOCK\_SIZE** when creating a general tablespace, **FILE\_BLOCK\_SIZE** defaults to [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size). When **FILE\_BLOCK\_SIZE** is equal to [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), the tablespace may only contain tables with an uncompressed row format (**COMPACT**, **REDUNDANT**, and **DYNAMIC** row formats).

##### Moving Tables Between Tablespaces Using ALTER TABLE

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) with the **TABLESPACE** option can be used to move a table to an existing general tablespace, to a new file-per-table tablespace, or to the system tablespace.

**Note**

Support for placing table partitions in shared tablespaces was deprecated in MySQL 5.7.24 and removed MySQL 8.0.13. Shared tablespaces include the **InnoDB** system tablespace and general tablespaces.

To move a table from a file-per-table tablespace or from the system tablespace to a general tablespace, specify the name of the general tablespace. The general tablespace must exist. See [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) for more information.

ALTER TABLE tbl\_name TABLESPACE [=] ***tablespace\_name***;

To move a table from a general tablespace or file-per-table tablespace to the system tablespace, specify **innodb\_system** as the tablespace name.

ALTER TABLE tbl\_name TABLESPACE [=] innodb\_system;

To move a table from the system tablespace or a general tablespace to a file-per-table tablespace, specify **innodb\_file\_per\_table** as the tablespace name.

ALTER TABLE tbl\_name TABLESPACE [=] innodb\_file\_per\_table;

**ALTER TABLE ... TABLESPACE** operations cause a full table rebuild, even if the **TABLESPACE** attribute has not changed from its previous value.

**ALTER TABLE ... TABLESPACE** syntax does not support moving a table from a temporary tablespace to a persistent tablespace.

The **DATA DIRECTORY** clause is permitted with **CREATE TABLE ... TABLESPACE=innodb\_file\_per\_table** but is otherwise not supported for use in combination with the **TABLESPACE** option. As of MySQL 8.0.21, the directory specified in a **DATA DIRECTORY** clause must be known to **InnoDB**. For more information, see [Using the DATA DIRECTORY Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-data-directory).

Restrictions apply when moving tables from encrypted tablespaces. See [Encryption Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-limitations).

##### Renaming a General Tablespace

Renaming a general tablespace is supported using [**ALTER TABLESPACE ... RENAME TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) syntax.

ALTER TABLESPACE s1 RENAME TO s2;

The [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-tablespace) privilege is required to rename a general tablespace.

**RENAME TO** operations are implicitly performed in [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode regardless of the [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) setting.

A **RENAME TO** operation cannot be performed while [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) or [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) is in effect for tables that reside in the tablespace.

Exclusive [metadata locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metadata_lock) are taken on tables within a general tablespace while the tablespace is renamed, which prevents concurrent DDL. Concurrent DML is supported.

##### Dropping a General Tablespace

The [**DROP TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-tablespace) statement is used to drop an **InnoDB** general tablespace.

All tables must be dropped from the tablespace prior to a [**DROP TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-tablespace) operation. If the tablespace is not empty, [**DROP TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-tablespace) returns an error.

Use a query similar to the following to identify tables in a general tablespace.

mysql> **SELECT a.NAME AS space\_name, b.NAME AS table\_name FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES a,**

**INFORMATION\_SCHEMA.INNODB\_TABLES b WHERE a.SPACE=b.SPACE AND a.NAME LIKE 'ts1';**

+------------+------------+

| space\_name | table\_name |

+------------+------------+

| ts1 | test/t1 |

| ts1 | test/t2 |

| ts1 | test/t3 |

+------------+------------+

A general **InnoDB** tablespace is not deleted automatically when the last table in the tablespace is dropped. The tablespace must be dropped explicitly using [**DROP TABLESPACE *tablespace\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-tablespace).

A general tablespace does not belong to any particular database. A [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) operation can drop tables that belong to a general tablespace but it cannot drop the tablespace, even if the [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) operation drops all tables that belong to the tablespace.

Similar to the system tablespace, truncating or dropping tables stored in a general tablespace creates free space internally in the general tablespace [.ibd data file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) which can only be used for new **InnoDB** data. Space is not released back to the operating system as it is when a file-per-table tablespace is deleted during a [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) operation.

This example demonstrates how to drop an **InnoDB** general tablespace. The general tablespace **ts1** is created with a single table. The table must be dropped before dropping the tablespace.

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' Engine=InnoDB;**

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1 Engine=InnoDB;**

mysql> **DROP TABLE t1;**

mysql> **DROP TABLESPACE ts1;**

**Note**

***tablespace\_name*** is a case-sensitive identifier in MySQL.

##### General Tablespace Limitations

A generated or existing tablespace cannot be changed to a general tablespace.

Creation of temporary general tablespaces is not supported.

General tablespaces do not support temporary tables.

Similar to the system tablespace, truncating or dropping tables stored in a general tablespace creates free space internally in the general tablespace [.ibd data file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) which can only be used for new **InnoDB** data. Space is not released back to the operating system as it is for [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces.

Additionally, a table-copying [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation on table that resides in a shared tablespace (a general tablespace or the system tablespace) can increase the amount of space used by the tablespace. Such operations require as much additional space as the data in the table plus indexes. The additional space required for the table-copying [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation is not released back to the operating system as it is for file-per-table tablespaces.

[**ALTER TABLE ... DISCARD TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**ALTER TABLE ...IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) are not supported for tables that belong to a general tablespace.

Support for placing table partitions in general tablespaces was deprecated in MySQL 5.7.24 and removed in MySQL 8.0.13.

The **ADD DATAFILE** clause is not supported in a replication environment where the source and replica reside on the same host, as it would cause the source and replica to create a tablespace of the same name in the same location, which is not supported. However, if the **ADD DATAFILE** clause is omitted, the tablespace is created in the data directory with a generated file name that is unique, which is permitted.

As of MySQL 8.0.21, general tablespaces cannot be created in the undo tablespace directory ([**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory)) unless that directly is known to **InnoDB**. Known directories are those defined by the [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), and [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variables.

#### 15.6.3.4 Undo Tablespaces

Undo tablespaces contain undo logs, which are collections of records containing information about how to undo the latest change by a transaction to a clustered index record.

Undo tablespaces are described under the following topics in this section:

[Default Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-default-undo-tablespaces)

[Undo Tablespace Size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb--undo-tablespace-size)

[Adding Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-add-undo-tablespaces)

[Dropping Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-drop-undo-tablespaces)

[Moving Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-move-undo-tablespaces)

[Configuring the Number of Rollback Segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespace-rollback-segments)

[Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace)

[Undo Tablespace Status Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespace-status-variables)

##### Default Undo Tablespaces

Two default undo tablespaces are created when the MySQL instance is initialized. Default undo tablespaces are created at initialization time to provide a location for rollback segments that must exist before SQL statements can be accepted. A minimum of two undo tablespaces is required to support automated truncation of undo tablespaces. See [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

Default undo tablespaces are created in the location defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable. If the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is undefined, default undo tablespaces are created in the data directory. Default undo tablespace data files are named undo\_001 and undo\_002. The corresponding undo tablespace names defined in the data dictionary are **innodb\_undo\_001** and **innodb\_undo\_002**.

As of MySQL 8.0.14, additional undo tablespaces can be created at runtime using SQL. See [Adding Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-add-undo-tablespaces).

##### Undo Tablespace Size

Prior to MySQL 8.0.23, the initial size of an undo tablespace depends on the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. For the default 16KB page size, the initial undo tablespace file size is 10MiB. For 4KB, 8KB, 32KB, and 64KB page sizes, the initial undo tablespace files sizes are 7MiB, 8MiB, 20MiB, and 40MiB, respectively. As of MySQL 8.0.23, the initial undo tablespace size is normally 16MiB. The initial size may differ when a new undo tablespace is created by a truncate operation. In this case, if the file extension size is larger than 16MB, and the previous file extension occurred within the last second, the new undo tablespace is created at a quarter of the size defined by the [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) variable.

Prior to MySQL 8.0.23, an undo tablespace is extended four extents at a time. From MySQL 8.0.23, an undo tablespace is extended by a minimum of 16MB. To handle aggressive growth, the file extension size is doubled if the previous file extension happened less than 0.1 seconds earlier. Doubling of the extension size can occur multiple times to a maximum of 256MB. If the previous file extension occurred more than 0.1 seconds earlier, the extension size is reduced by half, which can also occur multiple times, to a minimum of 16MB. If the **AUTOEXTEND\_SIZE** option is defined for an undo tablespace, it is extended by the greater of the **AUTOEXTEND\_SIZE** setting and the extension size determined by the logic described above. For information about the **AUTOEXTEND\_SIZE** option, see [Section 15.6.3.9, “Tablespace AUTOEXTEND\_SIZE Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size).

##### Adding Undo Tablespaces

Because undo logs can become large during long-running transactions, creating additional undo tablespaces can help prevent individual undo tablespaces from becoming too large. As of MySQL 8.0.14, additional undo tablespaces can be created at runtime using [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax.

CREATE UNDO TABLESPACE ***tablespace\_name*** ADD DATAFILE '***file\_name***.ibu';

The undo tablespace file name must have an .ibu extension. It is not permitted to specify a relative path when defining the undo tablespace file name. A fully qualified path is permitted, but the path must be known to **InnoDB**. Known paths are those defined by the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable. Unique undo tablespace file names are recommended to avoid potential file name conflicts when moving or cloning data.

**Note**

In a replication environment, the source and each replica must have its own undo tablespace file directory. Replicating the creation of an undo tablespace file to a common directory would cause a file name conflict.

At startup, directories defined by the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable are scanned for undo tablespace files. (The scan also traverses subdirectories.) Directories defined by the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) variables are automatically appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) value regardless of whether the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable is defined explicitly. An undo tablespace can therefore reside in paths defined by any of those variables.

If the undo tablespace file name does not include a path, the undo tablespace is created in the directory defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable. If that variable is undefined, the undo tablespace is created in the data directory.

**Note**

The **InnoDB** recovery process requires that undo tablespace files reside in known directories. Undo tablespace files must be discovered and opened before redo recovery and before other data files are opened to permit uncommitted transactions and data dictionary changes to be rolled back. An undo tablespace not found before recovery cannot be used, which can lead to database inconsistencies. An error message is reported at startup if an undo tablespace known to the data dictionary is not found. The known directory requirement also supports undo tablespace portability. See [Moving Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-move-undo-tablespaces).

To create undo tablespaces in a path relative to the data directory, set the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable to the relative path, and specify the file name only when creating an undo tablespace.

To view undo tablespace names and paths, query [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table):

SELECT TABLESPACE\_NAME, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES

WHERE FILE\_TYPE LIKE 'UNDO LOG';

A MySQL instance supports up to 127 undo tablespaces including the two default undo tablespaces created when the MySQL instance is initialized.

**Note**

Prior to MySQL 8.0.14, additional undo tablespaces are created by configuring the [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) startup variable. This variable is deprecated and no longer configurable as of MySQL 8.0.14.

Prior to MySQL 8.0.14, increasing the [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) setting creates the specified number of undo tablespaces and adds them to the list of active undo tablespaces. Decreasing the [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) setting removes undo tablespaces from the list of active undo tablespaces. Undo tablespaces that are removed from the active list remain active until they are no longer used by existing transactions. The [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) variable can be configured at runtime using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-statement) statement or defined in a configuration file.

Prior to MySQL 8.0.14, deactivated undo tablespaces cannot be removed. Manual removal of undo tablespace files is possible after a slow shutdown but is not recommended, as deactivated undo tablespaces may contain active undo logs for some time after the server is restarted if open transactions were present when shutting down the server. As of MySQL 8.0.14, undo tablespaces can be dropped using [**DROP UNDO TABALESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) syntax. See [Dropping Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-drop-undo-tablespaces).

##### Dropping Undo Tablespaces

As of MySQL 8.0.14, undo tablespaces created using [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax can be dropped at runtime using [**DROP UNDO TABALESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) syntax.

An undo tablespace must be empty before it can be dropped. To empty an undo tablespace, the undo tablespace must first be marked as inactive using [**ALTER UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) syntax so that the tablespace is no longer used for assigning rollback segments to new transactions.

ALTER UNDO TABLESPACE ***tablespace\_name*** SET INACTIVE;

After an undo tablespace is marked as inactive, transactions currently using rollback segments in the undo tablespace are permitted to finish, as are any transactions started before those transactions are completed. After transactions are completed, the purge system frees the rollback segments in the undo tablespace, and the undo tablespace is truncated to its initial size. (The same process is used when truncating undo tablespaces. See [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).) Once the undo tablespace is empty, it can be dropped.

DROP UNDO TABLESPACE ***tablespace\_name***;

**Note**

Alternatively, the undo tablespace can be left in an empty state and reactivated later, if needed, by issuing an [**ALTER UNDO TABLESPACE *tablespace\_name* SET ACTIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement.

The state of an undo tablespace can be monitored by querying the [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table.

SELECT NAME, STATE FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES

WHERE NAME LIKE '***tablespace\_name***';

An **inactive** state indicates that rollback segments in an undo tablespace are no longer used by new transactions. An **empty** state indicates that an undo tablespace is empty and ready to be dropped, or ready to be made active again using an [**ALTER UNDO TABLESPACE *tablespace\_name* SET ACTIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement. Attempting to drop an undo tablespace that is not empty returns an error.

The default undo tablespaces (**innodb\_undo\_001** and **innodb\_undo\_002**) created when the MySQL instance is initialized cannot be dropped. They can, however, be made inactive using an [**ALTER UNDO TABLESPACE *tablespace\_name* SET INACTIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement. Before a default undo tablespace can be made inactive, there must be an undo tablespace to take its place. A minimum of two active undo tablespaces are required at all times to support automated truncation of undo tablespaces.

##### Moving Undo Tablespaces

Undo tablespaces created with [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax can be moved while the server is offline to any known directory. Known directories are those defined by the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable. Directories defined by [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) are automatically appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) value regardless of whether the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable is defined explicitly. Those directories and their subdirectories are scanned at startup for undo tablespaces files. An undo tablespace file moved to any of those directories is discovered at startup and assumed to be the undo tablespace that was moved.

The default undo tablespaces (**innodb\_undo\_001** and **innodb\_undo\_002**) created when the MySQL instance is initialized must reside in the directory defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable. If the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is undefined, default undo tablespaces reside in the data directory. If default undo tablespaces are moved while the server is offline, the server must be started with the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable configured to the new directory.

The I/O patterns for undo logs make undo tablespaces good candidates for [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) storage.

##### Configuring the Number of Rollback Segments

The [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) variable defines the number of [rollback segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback_segment) allocated to each undo tablespace and to the global temporary tablespace. The [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) variable can be configured at startup or while the server is running.

The default setting for [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) is 128, which is also the maximum value. For information about the number of transactions that a rollback segment supports, see [Section 15.6.6, “Undo Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-logs).

##### Truncating Undo Tablespaces

There are two methods of truncating undo tablespaces, which can be used individually or in combination to manage undo tablespace size. One method is automated, enabled using configuration variables. The other method is manual, performed using SQL statements.

The automated method does not require monitoring undo tablespace size and, once enabled, it performs deactivation, truncation, and reactivation of undo tablespaces without manual intervention. The manual truncation method may be preferable if you want to control when undo tablespaces are taken offline for truncation. For example, you may want to avoid truncating undo tablespaces during peak workload times.

###### Automated Truncation

Automated truncation of undo tablespaces requires a minimum of two active undo tablespaces, which ensures that one undo tablespace remains active while the other is taken offline to be truncated. By default, two undo tablespaces are created when the MySQL instance is initialized.

To have undo tablespaces automatically truncated, enable the [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) variable. For example:

mysql> **SET GLOBAL innodb\_undo\_log\_truncate=ON;**

When the [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) variable is enabled, undo tablespaces that exceed the size limit defined by the [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) variable are subject to truncation. The [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) variable is dynamic and has a default value of 1073741824 bytes (1024 MiB).

mysql> **SELECT @@innodb\_max\_undo\_log\_size;**

+----------------------------+

| @@innodb\_max\_undo\_log\_size |

+----------------------------+

| 1073741824 |

+----------------------------+

When the [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) variable is enabled:

Default and user-defined undo tablespaces that exceed the [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) setting are marked for truncation. Selection of an undo tablespace for truncation is performed in a circular fashion to avoid truncating the same undo tablespace each time.

Rollback segments residing in the selected undo tablespace are made inactive so that they are not assigned to new transactions. Existing transactions that are currently using rollback segments are permitted to finish.

The [purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge) system empties rollback segments by freeing undo logs that are no longer in use.

After all rollback segments in the undo tablespace are freed, the truncate operation runs and truncates the undo tablespace to its initial size.

The size of an undo tablespace after a truncate operation may be larger than the initial size due to immediate use following the completion of the operation.

The [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable defines the location of default undo tablespace files. If the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is undefined, default undo tablespaces reside in the data directory. The location of all undo tablespace files including user-defined undo tablespaces created using [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax can be determined by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table:

SELECT TABLESPACE\_NAME, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES WHERE FILE\_TYPE LIKE 'UNDO LOG';

Rollback segments are reactivated so that they can be assigned to new transactions.

###### Manual Truncation

Manual truncation of undo tablespaces requires a minimum of three active undo tablespaces. Two active undo tablespaces are required at all times to support the possibility that automated truncation is enabled. A minimum of three undo tablespaces satisfies this requirement while permitting an undo tablespace to be taken offline manually.

To manually initiate truncation of an undo tablespace, deactivate the undo tablespace by issuing the following statement:

ALTER UNDO TABLESPACE ***tablespace\_name*** SET INACTIVE;

After the undo tablespace is marked as inactive, transactions currently using rollback segments in the undo tablespace are permitted to finish, as are any transactions started before those transactions are completed. After transactions are completed, the purge system frees the rollback segments in the undo tablespace, the undo tablespace is truncated to its initial size, and the undo tablespace state changes from **inactive** to **empty**.

**Note**

When an **ALTER UNDO TABLESPACE *tablespace\_name* SET INACTIVE** statement deactivates an undo tablespace, the purge thread looks for that undo tablespace at the next opportunity. Once the undo tablespace is found and marked for truncation, the purge thread returns with increased frequency to quickly empty and truncate the undo tablespace.

To check the state of an undo tablespace, query the [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table.

SELECT NAME, STATE FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES

WHERE NAME LIKE '***tablespace\_name***';

Once the undo tablespace is in an **empty** state, it can be reactivated by issuing the following statement:

ALTER UNDO TABLESPACE ***tablespace\_name*** SET ACTIVE;

An undo tablespace in an **empty** state can also be dropped. See [Dropping Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-drop-undo-tablespaces).

###### Expediting Automated Truncation of Undo Tablespaces

The purge thread is responsible for emptying and truncating undo tablespaces. By default, the purge thread looks for undo tablespaces to truncate once every 128 times that purge is invoked. The frequency with which the purge thread looks for undo tablespaces to truncate is controlled by the [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) variable, which has a default setting of 128.

mysql> **SELECT @@innodb\_purge\_rseg\_truncate\_frequency;**

+----------------------------------------+

| @@innodb\_purge\_rseg\_truncate\_frequency |

+----------------------------------------+

| 128 |

+----------------------------------------+

To increase the frequency, decrease the [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) setting. For example, to have the purge thread look for undo tabespaces once every 32 times that purge is invoked, set [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) to 32.

mysql> **SET GLOBAL innodb\_purge\_rseg\_truncate\_frequency=32;**

###### Performance Impact of Truncating Undo Tablespace Files

When an undo tablespace is truncated, the rollback segments in the undo tablespace are deactivated. The active rollback segments in other undo tablespaces assume responsibility for the entire system load, which may result in a slight performance degradation. The extent to which performance is affected depends on a number of factors:

Number of undo tablespaces

Number of undo logs

Undo tablespace size

Speed of the I/O susbsystem

Existing long running transactions

System load

The easiest way to avoid the potential performance impact is to increase the number of undo tablespaces.

###### Monitoring Undo Tablespace Truncation

As of MySQL 8.0.16, **undo** and **purge** susbsystem counters are provided for monitoring background activities associated with undo log truncation. For counter names and descriptions, query the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table.

SELECT NAME, SUBSYSTEM, COMMENT FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME LIKE '%truncate%';

For information about enabling counters and querying counter data, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

###### Undo Tablespace Truncation Limit

As of MySQL 8.0.21, the number of truncate operations on the same undo tablespace between checkpoints is limited to 64. The limit prevents potential issues caused by an excessive number of undo tablespace truncate operations, which can occur if [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) is set too low on a busy system, for example. If the limit is exceeded, an undo tablespace can still be made inactive, but it is not truncated until after the next checkpoint. The the limit was raised from 64 to 50,000 in MySQL 8.0.22.

###### Undo Tablespace Truncation Recovery

An undo tablespace truncate operation creates a temporary undo\_***space\_number***\_trunc.log file in the server log directory. That log directory is defined by [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir). If a system failure occurs during the truncate operation, the temporary log file permits the startup process to identify undo tablespaces that were being truncated and to continue the operation.

##### Undo Tablespace Status Variables

The following status variables permit tracking the total number of undo tablespaces, implicit (**InnoDB**-created) undo tablespaces, explicit (user-created) undo tablespaces, and the number of active undo tablespaces:

mysql> **SHOW STATUS LIKE 'Innodb\_undo\_tablespaces%';**

+----------------------------------+-------+

| Variable\_name | Value |

+----------------------------------+-------+

| Innodb\_undo\_tablespaces\_total | 2 |

| Innodb\_undo\_tablespaces\_implicit | 2 |

| Innodb\_undo\_tablespaces\_explicit | 0 |

| Innodb\_undo\_tablespaces\_active | 2 |

+----------------------------------+-------+

For status variable descriptions, see [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

#### 15.6.3.5 Temporary Tablespaces

**InnoDB** uses session temporary tablespaces and a global temporary tablespace.

##### Session Temporary Tablespaces

Session temporary tablespaces store user-created temporary tables and internal temporary tables created by the optimizer when **InnoDB** is configured as the storage engine for on-disk internal temporary tables. Beginning with MySQL 8.0.16, the storage engine used for on-disk internal temporary tables is always **InnoDB**. (Previously, the storage engine was determined by the value of [**internal\_tmp\_disk\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_internal_tmp_disk_storage_engine).)

Session temporary tablespaces are allocated to a session from a pool of temporary tablespaces on the first request to create an on-disk temporary table. A maximum of two tablespaces is allocated to a session, one for user-created temporary tables and the other for internal temporary tables created by the optimizer. The temporary tablespaces allocated to a session are used for all on-disk temporary tables created by the session. When a session disconnects, its temporary tablespaces are truncated and released back to the pool. A pool of 10 temporary tablespaces is created when the server is started. The size of the pool never shrinks and tablespaces are added to the pool automatically as necessary. The pool of temporary tablespaces is removed on normal shutdown or on an aborted initialization. Session temporary tablespace files are five pages in size when created and have an .ibt file name extension.

A range of 400 thousand space IDs is reserved for session temporary tablespaces. Because the pool of session temporary tablespaces is recreated each time the server is started, space IDs for session temporary tablespaces are not persisted when the server is shut down and may be reused.

The [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir) variable defines the location where session temporary tablespaces are created. The default location is the #innodb\_temp directory in the data directory. Startup is refused if the pool of temporary tablespaces cannot be created.

shell> cd ***BASEDIR***/data/#innodb\_temp

shell> ls

temp\_10.ibt temp\_2.ibt temp\_4.ibt temp\_6.ibt temp\_8.ibt

temp\_1.ibt temp\_3.ibt temp\_5.ibt temp\_7.ibt temp\_9.ibt

In statement based replication (SBR) mode, temporary tables created on a replica reside in a single session temporary tablespace that is truncated only when the MySQL server is shut down.

The [**INNODB\_SESSION\_TEMP\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-session-temp-tablespaces-table) table provides metadata about session temporary tablespaces.

The [**INFORMATION\_SCHEMA.INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) table provides metadata about user-created temporary tables that are active in an **InnoDB** instance.

##### Global Temporary Tablespace

The global temporary tablespace (ibtmp1) stores rollback segments for changes made to user-created temporary tables.

The [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) variable defines the relative path, name, size, and attributes for global temporary tablespace data files. If no value is specified for [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path), the default behavior is to create a single auto-extending data file named ibtmp1 in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory. The initial file size is slightly larger than 12MB.

The global temporary tablespace is removed on normal shutdown or on an aborted initialization, and recreated each time the server is started. The global temporary tablespace receives a dynamically generated space ID when it is created. Startup is refused if the global temporary tablespace cannot be created. The global temporary tablespace is not removed if the server halts unexpectedly. In this case, a database administrator can remove the global temporary tablespace manually or restart the MySQL server. Restarting the MySQL server removes and recreates the global temporary tablespace automatically.

The global temporary tablespace cannot reside on a raw device.

[**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) provides metadata about the global temporary tablespace. Issue a query similar to this one to view global temporary tablespace metadata:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.FILES WHERE TABLESPACE\_NAME='innodb\_temporary'\G**

By default, the global temporary tablespace data file is autoextending and increases in size as necessary.

To determine if a global temporary tablespace data file is autoextending, check the [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) setting:

mysql> **SELECT @@innodb\_temp\_data\_file\_path;**

+------------------------------+

| @@innodb\_temp\_data\_file\_path |

+------------------------------+

| ibtmp1:12M:autoextend |

+------------------------------+

To check the size of global temporary tablespace data files, query the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table using a query similar to this one:

mysql> **SELECT FILE\_NAME, TABLESPACE\_NAME, ENGINE, INITIAL\_SIZE, TOTAL\_EXTENTS\*EXTENT\_SIZE**

**AS TotalSizeBytes, DATA\_FREE, MAXIMUM\_SIZE FROM INFORMATION\_SCHEMA.FILES**

**WHERE TABLESPACE\_NAME = 'innodb\_temporary'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_NAME: ./ibtmp1

TABLESPACE\_NAME: innodb\_temporary

ENGINE: InnoDB

INITIAL\_SIZE: 12582912

TotalSizeBytes: 12582912

DATA\_FREE: 6291456

MAXIMUM\_SIZE: NULL

**TotalSizeBytes** shows the current size of the global temporary tablespace data file. For information about other field values, see [Section 26.3.15, “The INFORMATION\_SCHEMA FILES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table).

Alternatively, check the global temporary tablespace data file size on your operating system. The global temporary tablespace data file is located in the directory defined by the [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) variable.

To reclaim disk space occupied by a global temporary tablespace data file, restart the MySQL server. Restarting the server removes and recreates the global temporary tablespace data file according to the attributes defined by [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path).

To limit the size of the global temporary tablespace data file, configure [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) to specify a maximum file size. For example:

[mysqld]

innodb\_temp\_data\_file\_path=ibtmp1:12M:autoextend:max:500M

Configuring [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) requires restarting the server.

#### 15.6.3.6 Moving Tablespace Files While the Server is Offline

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) option, which defines directories to scan at startup for tablespace files, supports moving or restoring tablespace files to a new location while the server is offline. During startup, discovered tablespace files are used instead those referenced in the data dictionary, and the data dictionary is updated to reference the relocated files. If duplicate tablespace files are discovered by the scan, startup fails with an error indicating that multiple files were found for the same tablespace ID.

The directories defined by the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) configuration options are automatically appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value. These directories are scanned at startup regardless of whether the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) option is specified explicitly. The implicit addition of these directories permits moving system tablespace files, the data directory, or undo tablespace files without configuring the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting. However, settings must be updated when directories change. For example, after relocating the data directory, you must update the [**--datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) setting before restarting the server.

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) option may be specified in a startup command or MySQL option file. Quotes are used around the argument value because otherwise a semicolon (;) is interpreted as a special character by some command interpreters. (Unix shells treat it as a command terminator, for example.)

Startup command:

mysqld --innodb-directories="***directory\_path\_1***;***directory\_path\_2***"

MySQL option file:

[mysqld]

innodb\_directories="***directory\_path\_1***;***directory\_path\_2***"

The following procedure is applicable to moving individual [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) and [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) files, [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) files, [undo tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace) files, or the data directory. Before moving files or directories, review the usage notes that follow.

Stop the server.

Move the tablespace files or directories.

Make the new directory known to **InnoDB**.

If moving individual [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) or [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) files, add unknown directories to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) value.

The directories defined by the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) configuration options are automatically appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value, so you need not specify these.

A file-per-table tablespace file can only be moved to a directory with same name as the schema. For example, if the **actor** table belongs to the **sakila** schema, then the actor.ibd data file can only be moved to a directory named sakila.

General tablespace files cannot be moved to the data directory or a subdirectory of the data directory.

If moving system tablespace files, undo tablespaces, or the data directory, update the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) settings, as necessary.

Restart the server.

##### Usage Notes

Wildcard expressions cannot be used in the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value.

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) scan also traverses subdirectories of specified directories. Duplicate directories and subdirectories are discarded from the list of directories to be scanned.

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) option only supports moving **InnoDB** tablespace files. Moving files that belong to a storage engine other than **InnoDB** is not supported. This restriction also applies when moving the entire data directory.

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) option supports renaming of tablespace files when moving files to a scanned directory. It also supports moving tablespaces files to other supported operating systems.

When moving tablespace files to a different operating system, ensure that tablespace file names do not include prohibited characters or characters with a special meaning on the destination system.

When moving a data directory from a Windows operating system to a Linux operating system, modify the binary log file paths in the binary log index file to use backward slashes instead of forward slashes. By default, the binary log index file has the same base name as the binary log file, with the extension '.index'. The location of the binary log index file is defined by [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin). The default location is the data directory.

If moving tablespace files to a different operating system introduces cross-platform replication, it is the database administrator's responsibility to ensure proper replication of DDL statements that contain platform-specific directories. Statements that permit specifying directories include [**CREATE TABLE ... DATA DIRECTORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**CREATE TABLESPACE ... ADD DATAFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace).

Add the directories of file-per-table and general tablespaces created with an absolute path or in a location outside of the data directory to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting. Otherwise, **InnoDB** is not able to locate the files during recovery. For related information, see [Tablespace Discovery During Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-tablespace-discovery).

To view tablespace file locations, query the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table:

mysql> **SELECT TABLESPACE\_NAME, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES \G**

#### 15.6.3.7 Disabling Tablespace Path Validation

At startup, **InnoDB** scans directories defined by the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variable for tablespace files. The paths of discovered tablespace files are validated against the paths recorded in the data dictionary. If the paths do not match, the paths in the data dictionary are updated.

The [**innodb\_validate\_tablespace\_paths**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_validate_tablespace_paths) variable, introduced in MySQL 8.0.21, permits disabling tablespace path validation. This feature is intended for environments where tablespaces files are not moved. Disabling path validation improves startup time on systems with a large number of tablespace files. If [**log\_error\_verbosity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_log_error_verbosity) is set to 3, the following message is printed at startup when tablespace path validation is disabled:

[InnoDB] Skipping InnoDB tablespace path validation.

Manually moved tablespace files will not be detected!

**Warning**

Starting the server with tablespace path validation disabled after moving tablespace files can lead to undefined behavior.

#### 15.6.3.8 Optimizing Tablespace Space Allocation on Linux

As of MySQL 8.0.22, you can optimize how **InnoDB** allocates space to file-per-table and general tablespaces on Linux. By default, when additional space is required, **InnoDB** allocates pages to the tablespace and physically writes NULLs to those pages. This behavior can affect performance if new pages are allocated frequently. As of MySQL 8.0.22, you can disable [**innodb\_extend\_and\_initialize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_extend_and_initialize) on Linux systems to avoid physically writing NULLs to newly allocated tablespace pages. When [**innodb\_extend\_and\_initialize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_extend_and_initialize) is disabled, space is allocated to tablespace files using **posix\_fallocate()** calls, which reserve space without physically writing NULLs.

When pages are allocated using **posix\_fallocate()** calls, the extension size is small by default and pages are often allocated only a few at a time, which can cause fragmentation and increased random I/O. To avoid this potential issue, consider increasing the tablespace extension size when enabling **posix\_fallocate()** calls. Tablespace extension size can be increased up to 4GB using the **AUTOEXTEND\_SIZE** option. For more information, see [Section 15.6.3.9, “Tablespace AUTOEXTEND\_SIZE Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size).

**InnoDB** writes a redo log record before allocating a new tablespace page. If a page allocation operation is interrupted, the operation is replayed from the redo log record during recovery. (A page allocation operation replayed from a redo log record physically writes NULLs to the newly allocated page.) A redo log record is written before allocating a page regardless of the [**innodb\_extend\_and\_initialize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_extend_and_initialize) setting.

On non-Linux systems and Windows, **InnoDB** allocates new pages to the tablespace and physically writes NULLs to those pages, which is the default behavior. Attempting to disable [**innodb\_extend\_and\_initialize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_extend_and_initialize) on those systems returns the following error:

Changing innodb\_extend\_and\_initialize not supported on this platform. Falling back to the default.

#### 15.6.3.9 Tablespace AUTOEXTEND\_SIZE Configuration

By default, when a file-per-table or general tablespace requires additional space, the tablespace is extended incrementally according to the following rules:

If the tablespace is less than an extent in size, it is extended one page at a time.

If the tablespace is greater than 1 extent but smaller than 32 extents in size, it is extended one extent at a time.

If the tablespace is more than 32 extents in size, it is extended four extents at a time.

For information about extent size, see [Section 15.11.2, “File Space Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-space).

From MySQL 8.0.23, the amount by which a file-per-table or general tablespace is extended is configurable by specifying the **AUTOEXTEND\_SIZE** option. Configuring a larger extension size can help avoid fragmentation and facilitate ingestion of large amounts of data.

To configure the extension size for a file-per-table tablespace, specify the **AUTOEXTEND\_SIZE** size in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement:

CREATE TABLE t1 (c1 INT) AUTOEXTEND\_SIZE = 4M;

ALTER TABLE t1 AUTOEXTEND\_SIZE = 8M;

To configure the extension size for a general tablespace, specify the **AUTOEXTEND\_SIZE** size in a [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) or [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement:

CREATE TABLESPACE ts1 AUTOEXTEND\_SIZE = 4M;

ALTER TABLESPACE ts1 AUTOEXTEND\_SIZE = 8M;

**Note**

The **AUTOEXTEND\_SIZE** option can also be used when creating an undo tablespace, but the extension behavior for undo tablespaces differs. For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

The **AUTOEXTEND\_SIZE** setting must be a multiple of 4M. Specifying an **AUTOEXTEND\_SIZE** setting that is not a multiple of 4M returns an error.

The **AUTOEXTEND\_SIZE** default setting is 0, which causes the tablespace to be extended according to the default behavior described above.

The maximum **AUTOEXTEND\_SIZE** setting is 64M in MySQL 8.0.23. From MySQL 8.0.24, the maximum setting is 4GB.

The minimum **AUTOEXTEND\_SIZE** setting depends on the **InnoDB** page size, as shown in the following table:

| **InnoDB Page Size** | **Minimum AUTOEXTEND\_SIZE** |
| --- | --- |
| **4K** | **4M** |
| **8K** | **4M** |
| **16K** | **4M** |
| **32K** | **8M** |
| **64K** | **16M** |

The default **InnoDB** page size is 16K (16384 bytes). To determine the **InnoDB** page size for your MySQL instance, query the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) setting:

mysql> **SELECT @@GLOBAL.innodb\_page\_size;**

+---------------------------+

| @@GLOBAL.innodb\_page\_size |

+---------------------------+

| 16384 |

+---------------------------+

When the **AUTOEXTEND\_SIZE** setting for a tablespace is altered, the first extension that occurs afterward increases the tablespace size to a multiple of the **AUTOEXTEND\_SIZE** setting. Subsequent extensions are of the configured size.

When a file-per-table or general tablespace is created with a non-zero **AUTOEXTEND\_SIZE** setting, the tablespace is initialized at the specified **AUTOEXTEND\_SIZE** size.

[**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) cannot be used to configure the **AUTOEXTEND\_SIZE** of a file-per-table tablespace. [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) must be used.

For tables created in file-per-table tablespaces, [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) shows the **AUTOEXTEND\_SIZE** option only when it is configured to a non-zero value.

To determine the **AUTOEXTEND\_SIZE** for any **InnoDB** tablespace, query the [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table. For example:

mysql> **SELECT NAME, AUTOEXTEND\_SIZE FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES**

**WHERE NAME LIKE 'test/t1';**

+---------+-----------------+

| NAME | AUTOEXTEND\_SIZE |

+---------+-----------------+

| test/t1 | 4194304 |

+---------+-----------------+

mysql> **SELECT NAME, AUTOEXTEND\_SIZE FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES**

**WHERE NAME LIKE 'ts1';**

+------+-----------------+

| NAME | AUTOEXTEND\_SIZE |

+------+-----------------+

| ts1 | 4194304 |

+------+-----------------+

**Note**

An **AUTOEXTEND\_SIZE** of 0, which is the default setting, means that the tablespace is extended according to the default tablespace extension behavior described above.

### 15.6.4 Doublewrite Buffer

The doublewrite buffer is a storage area where **InnoDB** writes pages flushed from the buffer pool before writing the pages to their proper positions in the **InnoDB** data files. If there is an operating system, storage subsystem, or unexpected [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process exit in the middle of a page write, **InnoDB** can find a good copy of the page from the doublewrite buffer during crash recovery.

Although data is written twice, the doublewrite buffer does not require twice as much I/O overhead or twice as many I/O operations. Data is written to the doublewrite buffer in a large sequential chunk, with a single **fsync()** call to the operating system (except in the case that **innodb\_flush\_method** is set to **O\_DIRECT\_NO\_FSYNC**).

Prior to MySQL 8.0.20, the doublewrite buffer storage area is located in the **InnoDB** system tablespace. As of MySQL 8.0.20, the doublewrite buffer storage area is located in doublewrite files.

The following variables are provided for doublewrite buffer configuration:

[**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite)

The [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) variable controls whether the doublwrite buffer is enabled. It is enabled by default in most cases. To disable the doublewrite buffer, set [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) to 0 or start the server with **--skip-innodb-doublewrite**. You might consider disabling the doublewrite buffer if you are more concerned with performance than data integrity, as may be the case when performing benchmarks, for example.

If the doublewrite buffer is located on a Fusion-io device that supports atomic writes, the doublewrite buffer is automatically disabled and data file writes are performed using Fusion-io atomic writes instead. However, be aware that the [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) setting is global. When the doublewrite buffer is disabled, it is disabled for all data files including those that do not reside on Fusion-io hardware. This feature is only supported on Fusion-io hardware and is only enabled for Fusion-io NVMFS on Linux. To take full advantage of this feature, an [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) setting of **O\_DIRECT** is recommended.

[**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir)

The [**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir) variable (introduced in MySQL 8.0.20) defines the directory where **InnoDB** creates doublewrite files. If no directory is specified, doublewrite files are created in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory, which defaults to the data directory if unspecified.

A hash symbol '#' is automatically prefixed to the specified directory name to avoid conflicts with schema names. However, if a '.', '#'. or '/' prefix is specified explicitly in the directory name, the hash symbol '#' is not prefixed to the directory name.

Ideally, the doublewrite directory should be placed on the fastest storage media available.

[**innodb\_doublewrite\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_files)

The [**innodb\_doublewrite\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_files) variable defines the number of doublewrite files. By default, two doublewrite files are created for each buffer pool instance: A flush list doublewrite file and an LRU list doublewrite file.

The flush list doublewrite file is for pages flushed from the buffer pool flush list. The default size of a flush list doublewrite file is the **InnoDB** page size \* doublewrite page bytes.

The LRU list doublewrite file is for pages flushed from the buffer pool LRU list. It also contains slots for single page flushes. The default size of an LRU list doublewrite file is the **InnoDB** page size \* (doublewrite pages + (512 / the number of buffer pool instances)) where 512 is the total number of slots reserved for single page flushes.

At a minimum, there are two doublewrite files. The maximum number of doublewrite files is two times the number of buffer pool instances. (The number of buffer pool instances is controlled by the [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) variable.)

Doublewrite file names have the following format: #ib\_***page\_size***\_***file\_number***.dblwr. For example, the following doublewrite files are created for a MySQL instance with an **InnoDB** pages size of 16KB and a single buffer pool:

#ib\_16384\_0.dblwr

#ib\_16384\_1.dblwr

The [**innodb\_doublewrite\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_files) variable is intended for advanced performance tuning. The default setting should be suitable for most users.

[**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages)

The [**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages) variable (introduced in MySQL 8.0.20) controls the maximum number of doublewrite pages per thread. If no value is specified, [**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages) is set to the [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) value. This variable is intended for advanced performance tuning. The default value should be suitable for most users.

[**innodb\_doublewrite\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_batch_size)

The [**innodb\_doublewrite\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_batch_size) variable (introduced in MySQL 8.0.20) controls the number of doublewrite pages to write in a batch. This variable is intended for advanced performance tuning. The default value should be suitable for most users.

As of MySQL 8.0.23, **InnoDB** automatically encrypts doublewrite file pages that belong to encrypted tablespaces (see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption)). Likewise, doublewrite file pages belonging page-compressed tablespaces are compressed. As a result, doublewrite files can contain different page types including unencrypted and uncompressed pages, encrypted pages, compressed pages, and pages that are both encrypted and compressed.

### 15.6.5 Redo Log

The redo log is a disk-based data structure used during crash recovery to correct data written by incomplete transactions. During normal operations, the redo log encodes requests to change table data that result from SQL statements or low-level API calls. Modifications that did not finish updating the data files before an unexpected shutdown are replayed automatically during initialization, and before connections are accepted. For information about the role of the redo log in crash recovery, see [Section 15.18.2, “InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery).

By default, the redo log is physically represented on disk by two files named ib\_logfile0 and ib\_logfile1. MySQL writes to the redo log files in a circular fashion. Data in the redo log is encoded in terms of records affected; this data is collectively referred to as redo. The passage of data through the redo log is represented by an ever-increasing [LSN](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lsn) value.

Information and procedures related to redo logs are described under the following topics in the section:

[Changing the Number or Size of Redo Log Files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-redo-log-file-reconfigure)

[Group Commit for Redo Log Flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-group_commit)

[Redo Log Archiving](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-redo-log-archiving)

[Disabling Redo Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disable-redo-logging)

[Related Topics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-redo-log-related-topics)

#### Changing the Number or Size of Redo Log Files

To change the number or the size of [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) files, perform the following steps:

Stop the MySQL server and make sure that it shuts down without errors.

Edit my.cnf to change the log file configuration. To change the log file size, configure [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size). To increase the number of log files, configure [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group).

Start the MySQL server again.

If **InnoDB** detects that the [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) differs from the redo log file size, it writes a log checkpoint, closes and removes the old log files, creates new log files at the requested size, and opens the new log files.

#### Group Commit for Redo Log Flushing

**InnoDB**, like any other [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid)-compliant database engine, flushes the [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) of a transaction before it is committed. **InnoDB** uses [group commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_group_commit) functionality to group multiple flush requests together to avoid one flush for each commit. With group commit, **InnoDB** issues a single write to the log file to perform the commit action for multiple user transactions that commit at about the same time, significantly improving throughput.

#### Redo Log Archiving

Backup utilities that copy redo log records may sometimes fail to keep pace with redo log generation while a backup operation is in progress, resulting in lost redo log records due to those records being overwritten. This issue most often occurs when there is significant MySQL server activity during the backup operation, and the redo log file storage media operates at a faster speed than the backup storage media. The redo log archiving feature, introduced in MySQL 8.0.17, addresses this issue by sequentially writing redo log records to an archive file in addition to the redo log files. Backup utilities can copy redo log records from the archive file as necessary, thereby avoiding the potential loss of data.

If redo log archiving is configured on the server, [MySQL Enterprise Backup](https://dev.mysql.com/doc/mysql-enterprise-backup/8.0/en/), available with the [MySQL Enterprise Edition](https://www.mysql.com/products/enterprise/), uses the redo log archiving feature when backing up a MySQL server.

Enabling redo log archiving on the server requires setting a value for the [**innodb\_redo\_log\_archive\_dirs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_archive_dirs) system variable. The value is specified as a semicolon-separated list of labeled redo log archive directories. The ***label:directory*** pair is separated by a colon (**:**). For example:

mysql> SET GLOBAL innodb\_redo\_log\_archive\_dirs='***label1***:***directory\_path1***[;***label2***:***directory\_path2***;…]';

The ***label*** is an arbitrary identifier for the archive directory. It can be any string of characters, with the exception of colons (:), which are not permitted. An empty label is also permitted, but the colon (:) is still required in this case. A ***directory\_path*** must be specified. The directory selected for the redo log archive file must exist when redo log archiving is activated, or an error is returned. The path can contain colons (':'), but semicolons (;) are not permitted.

The [**innodb\_redo\_log\_archive\_dirs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_archive_dirs) variable must be configured before the redo log archiving can be activated. The default value is **NULL**, which does not permit activating redo log archiving.

**Notes**

The archive directories that you specify must satisfy the following requirements. (The requirements are enforced when redo log archiving is activated.):

Directories must exist. Directories are not created by the redo log archive process. Otherwise, the following error is returned:

ERROR 3844 (HY000): Redo log archive directory '***directory\_path1***' does not exist or is not a directory

Directories must not be world-accessible. This is to prevent the redo log data from being exposed to unauthorized users on the system. Otherwise, the following error is returned:

ERROR 3846 (HY000): Redo log archive directory '***directory\_path1***' is accessible to all OS users

Directories cannot be those defined by [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories), [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir), [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir), [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), or [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv), nor can they be parent directories or subdirectories of those directories. Otherwise, an error similar to the following is returned:

ERROR 3845 (HY000): Redo log archive directory '***directory\_path1***' is in, under, or over server directory 'datadir' - '***/path/to/data\_directory***'

When a backup utility that supports redo log archiving initiates a backup, the backup utility activates redo log archiving by invoking the **innodb\_redo\_log\_archive\_start()** user-defined function.

If you are not using a backup utility that supports redo log archiving, redo log archiving can also be activated manually, as shown:

mysql> SELECT innodb\_redo\_log\_archive\_start('***label***', '***subdir***');

+------------------------------------------+

| innodb\_redo\_log\_archive\_start('***label***') |

+------------------------------------------+

| 0 |

+------------------------------------------+

Or:

mysql> **DO innodb\_redo\_log\_archive\_start('*label*', '*subdir*');**

Query OK, 0 rows affected (0.09 sec)

**Note**

The MySQL session that activates redo log archiving (using **innodb\_redo\_log\_archive\_start()**) must remain open for the duration of the archiving. The same session must deactivate redo log archiving (using **innodb\_redo\_log\_archive\_stop()**). If the session is terminated before the redo log archiving is explicitly deactivated, the server deactivates redo log archiving implicitly and removes the redo log archive file.

where ***label*** is a label defined by [**innodb\_redo\_log\_archive\_dirs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_archive_dirs); **subdir** is an optional argument for specifying a subdirectory of the directory identified by ***label*** for saving the archive file; it must be a simple directory name (no slash (/), backslash (\), or colon (:) is permitted). **subdir** can be empty, null, or it can be left out.

Only users with the [**INNODB\_REDO\_LOG\_ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_innodb-redo-log-archive) privilege can activate redo log archiving by invoking **innodb\_redo\_log\_archive\_start()**, or deactivate it using **innodb\_redo\_log\_archive\_stop()**. The MySQL user running the backup utility or the MySQL user activating and deactivating redo log archiving manually must have this privilege.

The redo log archive file path is ***directory\_identified\_by\_label***/[***subdir***/]archive.***serverUUID***.000001.log, where ***directory\_identified\_by\_label*** is the archive directory identified by the ***label*** argument for **innodb\_redo\_log\_archive\_start()**. ***subdir*** is the optional argument used for **innodb\_redo\_log\_archive\_start()**.

For example, the full path and name for a redo log archive file appears similar to the following:

/***directory\_path***/***subdirectory***/archive.e71a47dc-61f8-11e9-a3cb-080027154b4d.000001.log

After the backup utility finishes copying **InnoDB** data files, it deactivates redo log archiving by calling the **innodb\_redo\_log\_archive\_stop()** user-defined function.

If you are not using a backup utility that supports redo log archiving, redo log archiving can also be deactivated manually, as shown:

mysql> **SELECT innodb\_redo\_log\_archive\_stop();**

+--------------------------------+

| innodb\_redo\_log\_archive\_stop() |

+--------------------------------+

| 0 |

+--------------------------------+

Or:

mysql> **DO innodb\_redo\_log\_archive\_stop();**

Query OK, 0 rows affected (0.01 sec)

After the stop function completes successfully, the backup utility looks for the relevant section of redo log data from the archive file and copies it into the backup.

After the backup utility finishes copying the redo log data and no longer needs the redo log archive file, it deletes the archive file.

Removal of the archive file is the responsibility of the backup utility in normal situations. However, if the redo log archiving operation quits unexpectedly before **innodb\_redo\_log\_archive\_stop()** is called, the MySQL server removes the file.

##### Performance Considerations

Activating redo log archiving typically has a minor performance cost due to the additional write activity.

On Unix and Unix-like operating systems, the performance impact is typically minor, assuming there is not a sustained high rate of updates. On Windows, the performance impact is typically a bit higher, assuming the same.

If there is a sustained high rate of updates and the redo log archive file is on the same storage media as the redo log files, the performance impact may be more significant due to compounded write activity.

If there is a sustained high rate of updates and the redo log archive file is on slower storage media than the redo log files, performance is impacted arbitrarily.

Writing to the redo log archive file does not impede normal transactional logging except in the case that the redo log archive file storage media operates at a much slower rate than the redo log file storage media, and there is a large backlog of persisted redo log blocks waiting to be written to the redo log archive file. In this case, the transactional logging rate is reduced to a level that can be managed by the slower storage media where the redo log archive file resides.

#### Disabling Redo Logging

As of MySQL 8.0.21, you can disable redo logging using the [**ALTER INSTANCE DISABLE INNODB REDO\_LOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) statement. This functionality is intended for loading data into a new MySQL instance. Disabling redo logging speeds up data loading by avoiding redo log writes and doublewrite buffering.

**Warning**

This feature is intended only for loading data into a new MySQL instance. Do not disable redo logging on a production system. It is permitted to shutdown and restart the server while redo logging is disabled, but an unexpected server stoppage while redo logging is disabled can cause data loss and instance corruption.

Attempting to restart the server after an unexpected server stoppage while redo logging is disabled is refused with the following error:

[ERROR] [MY-013578] [InnoDB] Server was killed when Innodb Redo

logging was disabled. Data files could be corrupt. You can try

to restart the database with innodb\_force\_recovery=6

In this case, initialize a new MySQL instance and start the data loading procedure again.

The [**INNODB\_REDO\_LOG\_ENABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_innodb-redo-log-enable) privilege is required to enable and disable redo logging.

The [**Innodb\_redo\_log\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_redo_log_enabled) status variable permits monitoring redo logging status.

Cloning operations and redo log archiving are not permitted while redo logging is disabled and vice versa.

An [**ALTER INSTANCE [ENABLE|DISABLE] INNODB REDO\_LOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operation requires an exclusive backup metadata lock, which prevents other [**ALTER INSTANCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operations from executing concurrently. Other [**ALTER INSTANCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operations must wait for the lock to be released before executing.

The following procedure demonstrates how to disable redo logging when loading data into a new MySQL instance.

On the new MySQL instance, grant the [**INNODB\_REDO\_LOG\_ENABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_innodb-redo-log-enable) privilege to the user account responsible for disabling redo logging.

mysql> GRANT INNODB\_REDO\_LOG\_ENABLE ON \*.\* to 'data\_load\_admin';

As the **data\_load\_admin** user, disable redo logging:

mysql> ALTER INSTANCE DISABLE INNODB REDO\_LOG;

Check the [**Innodb\_redo\_log\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_redo_log_enabled) status variable to ensure that redo logging is disabled.

mysql> **SHOW GLOBAL STATUS LIKE 'Innodb\_redo\_log\_enabled';**

+-------------------------+-------+

| Variable\_name | Value |

+-------------------------+-------+

| Innodb\_redo\_log\_enabled | OFF |

+-------------------------+-------+

Run the data load operation.

As the **data\_load\_admin** user, enable redo logging after the data load operation finishes:

mysql> ALTER INSTANCE ENABLE INNODB REDO\_LOG;

Check the [**Innodb\_redo\_log\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_redo_log_enabled) status variable to ensure that redo logging is enabled.

mysql> **SHOW GLOBAL STATUS LIKE 'Innodb\_redo\_log\_enabled';**

+-------------------------+-------+

| Variable\_name | Value |

+-------------------------+-------+

| Innodb\_redo\_log\_enabled | ON |

+-------------------------+-------+

#### Related Topics

[Redo Log File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-log-file-configuration)

[Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging)

[Redo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-redo-log)

### 15.6.6 Undo Logs

An undo log is a collection of undo log records associated with a single read-write transaction. An undo log record contains information about how to undo the latest change by a transaction to a [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index) record. If another transaction needs to see the original data as part of a consistent read operation, the unmodified data is retrieved from undo log records. Undo logs exist within [undo log segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_log_segment), which are contained within [rollback segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback_segment). Rollback segments reside in [undo tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace) and in the [global temporary tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_global_temporary_tablespace).

Undo logs that reside in the global temporary tablespace are used for transactions that modify data in user-defined temporary tables. These undo logs are not redo-logged, as they are not required for crash recovery. They are used only for rollback while the server is running. This type of undo log benefits performance by avoiding redo logging I/O.

For information about data-at-rest encryption for undo logs, see [Undo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-undo-log).

Each undo tablespace and the global temporary tablespace individually support a maximum of 128 rollback segments. The [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) variable defines the number of rollback segments.

The number of transactions that a rollback segment supports depends on the number of undo slots in the rollback segment and the number of undo logs required by each transaction. The number of undo slots in a rollback segment differs according to **InnoDB** page size.

| **InnoDB Page Size** | **Number of Undo Slots in a Rollback Segment (InnoDB Page Size / 16)** |
| --- | --- |
| **4096 (4KB)** | **256** |
| **8192 (8KB)** | **512** |
| **16384 (16KB)** | **1024** |
| **32768 (32KB)** | **2048** |
| **65536 (64KB)** | **4096** |

A transaction is assigned up to four undo logs, one for each of the following operation types:

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations on user-defined tables

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations on user-defined tables

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations on user-defined temporary tables

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations on user-defined temporary tables

Undo logs are assigned as needed. For example, a transaction that performs [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations on regular and temporary tables requires a full assignment of four undo logs. A transaction that performs only [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations on regular tables requires a single undo log.

A transaction that performs operations on regular tables is assigned undo logs from an assigned undo tablespace rollback segment. A transaction that performs operations on temporary tables is assigned undo logs from an assigned global temporary tablespace rollback segment.

An undo log assigned to a transaction remains attached to the transaction for its duration. For example, an undo log assigned to a transaction for an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operation on a regular table is used for all [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations on regular tables performed by that transaction.

Given the factors described above, the following formulas can be used to estimate the number of concurrent read-write transactions that **InnoDB** is capable of supporting.

**Note**

It is possible to encounter a concurrent transaction limit error before reaching the number of concurrent read-write transactions that **InnoDB** is capable of supporting. This occurs when a rollback segment assigned to a transaction runs out of undo slots. In such cases, try rerunning the transaction.

When transactions perform operations on temporary tables, the number of concurrent read-write transactions that **InnoDB** is capable of supporting is constrained by the number of rollback segments allocated to the global temporary tablespace, which is 128 by default.

If each transaction performs either an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) ***or*** an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation, the number of concurrent read-write transactions that **InnoDB** is capable of supporting is:

(innodb\_page\_size / 16) \* innodb\_rollback\_segments \* number of undo tablespaces

If each transaction performs an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) ***and*** an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation, the number of concurrent read-write transactions that **InnoDB** is capable of supporting is:

(innodb\_page\_size / 16 / 2) \* innodb\_rollback\_segments \* number of undo tablespaces

If each transaction performs an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operation on a temporary table, the number of concurrent read-write transactions that **InnoDB** is capable of supporting is:

(innodb\_page\_size / 16) \* innodb\_rollback\_segments

If each transaction performs an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) ***and*** an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation on a temporary table, the number of concurrent read-write transactions that **InnoDB** is capable of supporting is:

(innodb\_page\_size / 16 / 2) \* innodb\_rollback\_segments

## 15.7 InnoDB Locking and Transaction Model

[15.7.1 InnoDB Locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking)

[15.7.2 InnoDB Transaction Model](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-model)

[15.7.3 Locks Set by Different SQL Statements in InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locks-set)

[15.7.4 Phantom Rows](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-next-key-locking)

[15.7.5 Deadlocks in InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks)

[15.7.6 Transaction Scheduling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-scheduling)

To implement a large-scale, busy, or highly reliable database application, to port substantial code from a different database system, or to tune MySQL performance, it is important to understand **InnoDB** locking and the **InnoDB** transaction model.

This section discusses several topics related to **InnoDB** locking and the **InnoDB** transaction model with which you should be familiar.

[Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking) describes lock types used by **InnoDB**.

[Section 15.7.2, “InnoDB Transaction Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-model) describes transaction isolation levels and the locking strategies used by each. It also discusses the use of [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), consistent non-locking reads, and locking reads.

[Section 15.7.3, “Locks Set by Different SQL Statements in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locks-set) discusses specific types of locks set in **InnoDB** for various statements.

[Section 15.7.4, “Phantom Rows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-next-key-locking) describes how **InnoDB** uses next-key locking to avoid phantom rows.

[Section 15.7.5, “Deadlocks in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks) provides a deadlock example, discusses deadlock detection, and provides tips for minimizing and handling deadlocks in **InnoDB**.

### 15.7.1 InnoDB Locking

This section describes lock types used by **InnoDB**.

[Shared and Exclusive Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-shared-exclusive-locks)

[Intention Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-intention-locks)

[Record Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-record-locks)

[Gap Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-gap-locks)

[Next-Key Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-next-key-locks)

[Insert Intention Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-insert-intention-locks)

[AUTO-INC Locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-inc-locks)

[Predicate Locks for Spatial Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-predicate-locks)

#### Shared and Exclusive Locks

**InnoDB** implements standard row-level locking where there are two types of locks, [shared (**S**) locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_shared_lock) and [exclusive (**X**) locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_exclusive_lock).

A [shared (**S**) lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_shared_lock) permits the transaction that holds the lock to read a row.

An [exclusive (**X**) lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_exclusive_lock) permits the transaction that holds the lock to update or delete a row.

If transaction **T1** holds a shared (**S**) lock on row **r**, then requests from some distinct transaction **T2** for a lock on row **r** are handled as follows:

A request by **T2** for an **S** lock can be granted immediately. As a result, both **T1** and **T2** hold an **S** lock on **r**.

A request by **T2** for an **X** lock cannot be granted immediately.

If a transaction **T1** holds an exclusive (**X**) lock on row **r**, a request from some distinct transaction **T2** for a lock of either type on **r** cannot be granted immediately. Instead, transaction **T2** has to wait for transaction **T1** to release its lock on row **r**.

#### Intention Locks

**InnoDB** supports multiple granularity locking which permits coexistence of row locks and table locks. For example, a statement such as [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) takes an exclusive lock (an **X** lock) on the specified table. To make locking at multiple granularity levels practical, **InnoDB** uses [intention locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_intention_lock). Intention locks are table-level locks that indicate which type of lock (shared or exclusive) a transaction requires later for a row in a table. There are two types of intention locks:

An [intention shared lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_intention_shared_lock) (**IS**) indicates that a transaction intends to set a shared lock on individual rows in a table.

An [intention exclusive lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_intention_exclusive_lock) (**IX**) indicates that a transaction intends to set an exclusive lock on individual rows in a table.

For example, [**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) sets an **IS** lock, and [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) sets an **IX** lock.

The intention locking protocol is as follows:

Before a transaction can acquire a shared lock on a row in a table, it must first acquire an **IS** lock or stronger on the table.

Before a transaction can acquire an exclusive lock on a row in a table, it must first acquire an **IX** lock on the table.

Table-level lock type compatibility is summarized in the following matrix.

|  | **X** | **IX** | **S** | **IS** |
| --- | --- | --- | --- | --- |
| **X** | Conflict | Conflict | Conflict | Conflict |
| **IX** | Conflict | Compatible | Conflict | Compatible |
| **S** | Conflict | Conflict | Compatible | Compatible |
| **IS** | Conflict | Compatible | Compatible | Compatible |

A lock is granted to a requesting transaction if it is compatible with existing locks, but not if it conflicts with existing locks. A transaction waits until the conflicting existing lock is released. If a lock request conflicts with an existing lock and cannot be granted because it would cause [deadlock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock), an error occurs.

Intention locks do not block anything except full table requests (for example, [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)). The main purpose of intention locks is to show that someone is locking a row, or going to lock a row in the table.

Transaction data for an intention lock appears similar to the following in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) and [InnoDB monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor) output:

TABLE LOCK table `test`.`t` trx id 10080 lock mode IX

#### Record Locks

A record lock is a lock on an index record. For example, **SELECT c1 FROM t WHERE c1 = 10 FOR UPDATE;** prevents any other transaction from inserting, updating, or deleting rows where the value of **t.c1** is **10**.

Record locks always lock index records, even if a table is defined with no indexes. For such cases, **InnoDB** creates a hidden clustered index and uses this index for record locking. See [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types).

Transaction data for a record lock appears similar to the following in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) and [InnoDB monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor) output:

RECORD LOCKS space id 58 page no 3 n bits 72 index `PRIMARY` of table `test`.`t`

trx id 10078 lock\_mode X locks rec but not gap

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 3; compact format; info bits 0

0: len 4; hex 8000000a; asc ;;

1: len 6; hex 00000000274f; asc 'O;;

2: len 7; hex b60000019d0110; asc ;;

#### Gap Locks

A gap lock is a lock on a gap between index records, or a lock on the gap before the first or after the last index record. For example, **SELECT c1 FROM t WHERE c1 BETWEEN 10 and 20 FOR UPDATE;** prevents other transactions from inserting a value of **15** into column **t.c1**, whether or not there was already any such value in the column, because the gaps between all existing values in the range are locked.

A gap might span a single index value, multiple index values, or even be empty.

Gap locks are part of the tradeoff between performance and concurrency, and are used in some transaction isolation levels and not others.

Gap locking is not needed for statements that lock rows using a unique index to search for a unique row. (This does not include the case that the search condition includes only some columns of a multiple-column unique index; in that case, gap locking does occur.) For example, if the **id** column has a unique index, the following statement uses only an index-record lock for the row having **id** value 100 and it does not matter whether other sessions insert rows in the preceding gap:

SELECT \* FROM child WHERE id = 100;

If **id** is not indexed or has a nonunique index, the statement does lock the preceding gap.

It is also worth noting here that conflicting locks can be held on a gap by different transactions. For example, transaction A can hold a shared gap lock (gap S-lock) on a gap while transaction B holds an exclusive gap lock (gap X-lock) on the same gap. The reason conflicting gap locks are allowed is that if a record is purged from an index, the gap locks held on the record by different transactions must be merged.

Gap locks in **InnoDB** are “purely inhibitive”, which means that their only purpose is to prevent other transactions from inserting to the gap. Gap locks can co-exist. A gap lock taken by one transaction does not prevent another transaction from taking a gap lock on the same gap. There is no difference between shared and exclusive gap locks. They do not conflict with each other, and they perform the same function.

Gap locking can be disabled explicitly. This occurs if you change the transaction isolation level to [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed). Under these circumstances, gap locking is disabled for searches and index scans and is used only for foreign-key constraint checking and duplicate-key checking.

There are also other effects of using the [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level. Record locks for nonmatching rows are released after MySQL has evaluated the **WHERE** condition. For **UPDATE** statements, **InnoDB** does a “semi-consistent” read, such that it returns the latest committed version to MySQL so that MySQL can determine whether the row matches the **WHERE** condition of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update).

#### Next-Key Locks

A next-key lock is a combination of a record lock on the index record and a gap lock on the gap before the index record.

**InnoDB** performs row-level locking in such a way that when it searches or scans a table index, it sets shared or exclusive locks on the index records it encounters. Thus, the row-level locks are actually index-record locks. A next-key lock on an index record also affects the “gap” before that index record. That is, a next-key lock is an index-record lock plus a gap lock on the gap preceding the index record. If one session has a shared or exclusive lock on record **R** in an index, another session cannot insert a new index record in the gap immediately before **R** in the index order.

Suppose that an index contains the values 10, 11, 13, and 20. The possible next-key locks for this index cover the following intervals, where a round bracket denotes exclusion of the interval endpoint and a square bracket denotes inclusion of the endpoint:

(negative infinity, 10]

(10, 11]

(11, 13]

(13, 20]

(20, positive infinity)

For the last interval, the next-key lock locks the gap above the largest value in the index and the “supremum” pseudo-record having a value higher than any value actually in the index. The supremum is not a real index record, so, in effect, this next-key lock locks only the gap following the largest index value.

By default, **InnoDB** operates in [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) transaction isolation level. In this case, **InnoDB** uses next-key locks for searches and index scans, which prevents phantom rows (see [Section 15.7.4, “Phantom Rows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-next-key-locking)).

Transaction data for a next-key lock appears similar to the following in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) and [InnoDB monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor) output:

RECORD LOCKS space id 58 page no 3 n bits 72 index `PRIMARY` of table `test`.`t`

trx id 10080 lock\_mode X

Record lock, heap no 1 PHYSICAL RECORD: n\_fields 1; compact format; info bits 0

0: len 8; hex 73757072656d756d; asc supremum;;

Record lock, heap no 2 PHYSICAL RECORD: n\_fields 3; compact format; info bits 0

0: len 4; hex 8000000a; asc ;;

1: len 6; hex 00000000274f; asc 'O;;

2: len 7; hex b60000019d0110; asc ;;

#### Insert Intention Locks

An insert intention lock is a type of gap lock set by [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations prior to row insertion. This lock signals the intent to insert in such a way that multiple transactions inserting into the same index gap need not wait for each other if they are not inserting at the same position within the gap. Suppose that there are index records with values of 4 and 7. Separate transactions that attempt to insert values of 5 and 6, respectively, each lock the gap between 4 and 7 with insert intention locks prior to obtaining the exclusive lock on the inserted row, but do not block each other because the rows are nonconflicting.

The following example demonstrates a transaction taking an insert intention lock prior to obtaining an exclusive lock on the inserted record. The example involves two clients, A and B.

Client A creates a table containing two index records (90 and 102) and then starts a transaction that places an exclusive lock on index records with an ID greater than 100. The exclusive lock includes a gap lock before record 102:

mysql> **CREATE TABLE child (id int(11) NOT NULL, PRIMARY KEY(id)) ENGINE=InnoDB;**

mysql> **INSERT INTO child (id) values (90),(102);**

mysql> **START TRANSACTION;**

mysql> **SELECT \* FROM child WHERE id > 100 FOR UPDATE;**

+-----+

| id |

+-----+

| 102 |

+-----+

Client B begins a transaction to insert a record into the gap. The transaction takes an insert intention lock while it waits to obtain an exclusive lock.

mysql> **START TRANSACTION;**

mysql> **INSERT INTO child (id) VALUES (101);**

Transaction data for an insert intention lock appears similar to the following in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) and [InnoDB monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor) output:

RECORD LOCKS space id 31 page no 3 n bits 72 index `PRIMARY` of table `test`.`child`

trx id 8731 lock\_mode X locks gap before rec ***insert intention*** waiting

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 3; compact format; info bits 0

0: len 4; hex 80000066; asc f;;

1: len 6; hex 000000002215; asc " ;;

2: len 7; hex 9000000172011c; asc r ;;...

#### AUTO-INC Locks

An **AUTO-INC** lock is a special table-level lock taken by transactions inserting into tables with **AUTO\_INCREMENT** columns. In the simplest case, if one transaction is inserting values into the table, any other transactions must wait to do their own inserts into that table, so that rows inserted by the first transaction receive consecutive primary key values.

The [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) configuration option controls the algorithm used for auto-increment locking. It allows you to choose how to trade off between predictable sequences of auto-increment values and maximum concurrency for insert operations.

For more information, see [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling).

#### Predicate Locks for Spatial Indexes

**InnoDB** supports **SPATIAL** indexing of columns containing spatial columns (see [Section 11.4.9, “Optimizing Spatial Analysis”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#optimizing-spatial-analysis)).

To handle locking for operations involving **SPATIAL** indexes, next-key locking does not work well to support [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) or [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) transaction isolation levels. There is no absolute ordering concept in multidimensional data, so it is not clear which is the “next” key.

To enable support of isolation levels for tables with **SPATIAL** indexes, **InnoDB** uses predicate locks. A **SPATIAL** index contains minimum bounding rectangle (MBR) values, so **InnoDB** enforces consistent read on the index by setting a predicate lock on the MBR value used for a query. Other transactions cannot insert or modify a row that would match the query condition.

### 15.7.2 InnoDB Transaction Model

[15.7.2.1 Transaction Isolation Levels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-isolation-levels)

[15.7.2.2 autocommit, Commit, and Rollback](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-autocommit-commit-rollback)

[15.7.2.3 Consistent Nonlocking Reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-consistent-read)

[15.7.2.4 Locking Reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads)

In the **InnoDB** transaction model, the goal is to combine the best properties of a [multi-versioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mvcc) database with traditional two-phase locking. **InnoDB** performs locking at the row level and runs queries as nonlocking [consistent reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_consistent_read) by default, in the style of Oracle. The lock information in **InnoDB** is stored space-efficiently so that lock escalation is not needed. Typically, several users are permitted to lock every row in **InnoDB** tables, or any random subset of the rows, without causing **InnoDB** memory exhaustion.

#### 15.7.2.1 Transaction Isolation Levels

Transaction isolation is one of the foundations of database processing. Isolation is the I in the acronym [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid); the isolation level is the setting that fine-tunes the balance between performance and reliability, consistency, and reproducibility of results when multiple transactions are making changes and performing queries at the same time.

**InnoDB** offers all four transaction isolation levels described by the SQL:1992 standard: [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted), [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read), and [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable). The default isolation level for **InnoDB** is [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read).

A user can change the isolation level for a single session or for all subsequent connections with the [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement. To set the server's default isolation level for all connections, use the [--transaction-isolation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_transaction-isolation) option on the command line or in an option file. For detailed information about isolation levels and level-setting syntax, see [Section 13.3.7, “SET TRANSACTION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction).

**InnoDB** supports each of the transaction isolation levels described here using different [locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking) strategies. You can enforce a high degree of consistency with the default [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) level, for operations on crucial data where [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid) compliance is important. Or you can relax the consistency rules with [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) or even [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted), in situations such as bulk reporting where precise consistency and repeatable results are less important than minimizing the amount of overhead for locking. [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) enforces even stricter rules than [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read), and is used mainly in specialized situations, such as with [XA](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_xa) transactions and for troubleshooting issues with concurrency and [deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock).

The following list describes how MySQL supports the different transaction levels. The list goes from the most commonly used level to the least used.

**REPEATABLE READ**

This is the default isolation level for **InnoDB**. [Consistent reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_consistent_read) within the same transaction read the [snapshot](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_snapshot) established by the first read. This means that if you issue several plain (nonlocking) [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements within the same transaction, these [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements are consistent also with respect to each other. See [Section 15.7.2.3, “Consistent Nonlocking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-consistent-read).

For [locking reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read) ([**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) with **FOR UPDATE** or **FOR SHARE**), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements, locking depends on whether the statement uses a unique index with a unique search condition, or a range-type search condition.

For a unique index with a unique search condition, **InnoDB** locks only the index record found, not the [gap](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap) before it.

For other search conditions, **InnoDB** locks the index range scanned, using [gap locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap_lock) or [next-key locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_next_key_lock) to block insertions by other sessions into the gaps covered by the range. For information about gap locks and next-key locks, see [Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking).

**READ COMMITTED**

Each consistent read, even within the same transaction, sets and reads its own fresh snapshot. For information about consistent reads, see [Section 15.7.2.3, “Consistent Nonlocking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-consistent-read).

For locking reads ([**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) with **FOR UPDATE** or **FOR SHARE**), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements, **InnoDB** locks only index records, not the gaps before them, and thus permits the free insertion of new records next to locked records. Gap locking is only used for foreign-key constraint checking and duplicate-key checking.

Because gap locking is disabled, phantom problems may occur, as other sessions can insert new rows into the gaps. For information about phantoms, see [Section 15.7.4, “Phantom Rows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-next-key-locking).

Only row-based binary logging is supported with the **READ COMMITTED** isolation level. If you use **READ COMMITTED** with [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format), the server automatically uses row-based logging.

Using **READ COMMITTED** has additional effects:

For [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements, **InnoDB** holds locks only for rows that it updates or deletes. Record locks for nonmatching rows are released after MySQL has evaluated the **WHERE** condition. This greatly reduces the probability of deadlocks, but they can still happen.

For [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, if a row is already locked, **InnoDB** performs a “semi-consistent” read, returning the latest committed version to MySQL so that MySQL can determine whether the row matches the **WHERE** condition of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). If the row matches (must be updated), MySQL reads the row again and this time **InnoDB** either locks it or waits for a lock on it.

Consider the following example, beginning with this table:

CREATE TABLE t (a INT NOT NULL, b INT) ENGINE = InnoDB;

INSERT INTO t VALUES (1,2),(2,3),(3,2),(4,3),(5,2);

COMMIT;

In this case, the table has no indexes, so searches and index scans use the hidden clustered index for record locking (see [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types)) rather than indexed columns.

Suppose that one session performs an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) using these statements:

# Session A

START TRANSACTION;

UPDATE t SET b = 5 WHERE b = 3;

Suppose also that a second session performs an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) by executing these statements following those of the first session:

# Session B

UPDATE t SET b = 4 WHERE b = 2;

As [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) executes each [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), it first acquires an exclusive lock for each row, and then determines whether to modify it. If [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) does not modify the row, it releases the lock. Otherwise, [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) retains the lock until the end of the transaction. This affects transaction processing as follows.

When using the default **REPEATABLE READ** isolation level, the first [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) acquires an x-lock on each row that it reads and does not release any of them:

x-lock(1,2); retain x-lock

x-lock(2,3); update(2,3) to (2,5); retain x-lock

x-lock(3,2); retain x-lock

x-lock(4,3); update(4,3) to (4,5); retain x-lock

x-lock(5,2); retain x-lock

The second [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) blocks as soon as it tries to acquire any locks (because first update has retained locks on all rows), and does not proceed until the first [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) commits or rolls back:

x-lock(1,2); block and wait for first UPDATE to commit or roll back

If **READ COMMITTED** is used instead, the first [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) acquires an x-lock on each row that it reads and releases those for rows that it does not modify:

x-lock(1,2); unlock(1,2)

x-lock(2,3); update(2,3) to (2,5); retain x-lock

x-lock(3,2); unlock(3,2)

x-lock(4,3); update(4,3) to (4,5); retain x-lock

x-lock(5,2); unlock(5,2)

For the second **UPDATE**, **InnoDB** does a “semi-consistent” read, returning the latest committed version of each row that it reads to MySQL so that MySQL can determine whether the row matches the **WHERE** condition of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update):

x-lock(1,2); update(1,2) to (1,4); retain x-lock

x-lock(2,3); unlock(2,3)

x-lock(3,2); update(3,2) to (3,4); retain x-lock

x-lock(4,3); unlock(4,3)

x-lock(5,2); update(5,2) to (5,4); retain x-lock

However, if the **WHERE** condition includes an indexed column, and **InnoDB** uses the index, only the indexed column is considered when taking and retaining record locks. In the following example, the first [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) takes and retains an x-lock on each row where b = 2. The second [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) blocks when it tries to acquire x-locks on the same records, as it also uses the index defined on column b.

CREATE TABLE t (a INT NOT NULL, b INT, c INT, INDEX (b)) ENGINE = InnoDB;

INSERT INTO t VALUES (1,2,3),(2,2,4);

COMMIT;

# Session A

START TRANSACTION;

UPDATE t SET b = 3 WHERE b = 2 AND c = 3;

# Session B

UPDATE t SET b = 4 WHERE b = 2 AND c = 4;

The **READ COMMITTED** isolation level can be set at startup or changed at runtime. At runtime, it can be set globally for all sessions, or individually per session.

**READ UNCOMMITTED**

**[SELECT](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "select" \o "13.2.10 SELECT Statement)** statements are performed in a nonlocking fashion, but a possible earlier version of a row might be used. Thus, using this isolation level, such reads are not consistent. This is also called a [dirty read](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_read). Otherwise, this isolation level works like [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed).

**SERIALIZABLE**

This level is like [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read), but **InnoDB** implicitly converts all plain [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements to [**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) if [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is disabled. If [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is enabled, the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is its own transaction. It therefore is known to be read only and can be serialized if performed as a consistent (nonlocking) read and need not block for other transactions. (To force a plain [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) to block if other transactions have modified the selected rows, disable [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit).)

**Note**

As of MySQL 8.0.22, DML operations that read data from MySQL grant tables (through a join list or subquery) but do not modify them do not acquire read locks on the MySQL grant tables, regardless of the isolation level. For more information, see [Grant Table Concurrency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#grant-tables-concurrency).

#### 15.7.2.2 autocommit, Commit, and Rollback

In **InnoDB**, all user activity occurs inside a transaction. If [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode is enabled, each SQL statement forms a single transaction on its own. By default, MySQL starts the session for each new connection with [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) enabled, so MySQL does a commit after each SQL statement if that statement did not return an error. If a statement returns an error, the commit or rollback behavior depends on the error. See [Section 15.21.4, “InnoDB Error Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-error-handling).

A session that has [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) enabled can perform a multiple-statement transaction by starting it with an explicit [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement and ending it with a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement. See [Section 13.3.1, “START TRANSACTION, COMMIT, and ROLLBACK Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

If [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode is disabled within a session with **SET autocommit = 0**, the session always has a transaction open. A [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement ends the current transaction and a new one starts.

If a session that has [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) disabled ends without explicitly committing the final transaction, MySQL rolls back that transaction.

Some statements implicitly end a transaction, as if you had done a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) before executing the statement. For details, see [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

A [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) means that the changes made in the current transaction are made permanent and become visible to other sessions. A [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement, on the other hand, cancels all modifications made by the current transaction. Both [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) and [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) release all **InnoDB** locks that were set during the current transaction.

##### Grouping DML Operations with Transactions

By default, connection to the MySQL server begins with [autocommit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_autocommit) mode enabled, which automatically commits every SQL statement as you execute it. This mode of operation might be unfamiliar if you have experience with other database systems, where it is standard practice to issue a sequence of [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) statements and commit them or roll them back all together.

To use multiple-statement [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction), switch autocommit off with the SQL statement **SET autocommit = 0** and end each transaction with [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) as appropriate. To leave autocommit on, begin each transaction with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) and end it with [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). The following example shows two transactions. The first is committed; the second is rolled back.

shell> **mysql test**

mysql> **CREATE TABLE customer (a INT, b CHAR (20), INDEX (a));**

Query OK, 0 rows affected (0.00 sec)

mysql> **-- Do a transaction with autocommit turned on.**

mysql> **START TRANSACTION;**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO customer VALUES (10, 'Heikki');**

Query OK, 1 row affected (0.00 sec)

mysql> **COMMIT;**

Query OK, 0 rows affected (0.00 sec)

mysql> **-- Do another transaction with autocommit turned off.**

mysql> **SET autocommit=0;**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO customer VALUES (15, 'John');**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO customer VALUES (20, 'Paul');**

Query OK, 1 row affected (0.00 sec)

mysql> **DELETE FROM customer WHERE b = 'Heikki';**

Query OK, 1 row affected (0.00 sec)

mysql> **-- Now we undo those last 2 inserts and the delete.**

mysql> **ROLLBACK;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT \* FROM customer;**

+------+--------+

| a | b |

+------+--------+

| 10 | Heikki |

+------+--------+

1 row in set (0.00 sec)

mysql>

###### Transactions in Client-Side Languages

In APIs such as PHP, Perl DBI, JDBC, ODBC, or the standard C call interface of MySQL, you can send transaction control statements such as [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to the MySQL server as strings just like any other SQL statements such as [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert). Some APIs also offer separate special transaction commit and rollback functions or methods.

#### 15.7.2.3 Consistent Nonlocking Reads

A [consistent read](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_consistent_read) means that **InnoDB** uses multi-versioning to present to a query a snapshot of the database at a point in time. The query sees the changes made by transactions that committed before that point of time, and no changes made by later or uncommitted transactions. The exception to this rule is that the query sees the changes made by earlier statements within the same transaction. This exception causes the following anomaly: If you update some rows in a table, a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) sees the latest version of the updated rows, but it might also see older versions of any rows. If other sessions simultaneously update the same table, the anomaly means that you might see the table in a state that never existed in the database.

If the transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) is [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) (the default level), all consistent reads within the same transaction read the snapshot established by the first such read in that transaction. You can get a fresher snapshot for your queries by committing the current transaction and after that issuing new queries.

With [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level, each consistent read within a transaction sets and reads its own fresh snapshot.

Consistent read is the default mode in which **InnoDB** processes [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements in [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) and [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) isolation levels. A consistent read does not set any locks on the tables it accesses, and therefore other sessions are free to modify those tables at the same time a consistent read is being performed on the table.

Suppose that you are running in the default [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) isolation level. When you issue a consistent read (that is, an ordinary [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement), **InnoDB** gives your transaction a timepoint according to which your query sees the database. If another transaction deletes a row and commits after your timepoint was assigned, you do not see the row as having been deleted. Inserts and updates are treated similarly.

**Note**

The snapshot of the database state applies to [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements within a transaction, not necessarily to [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) statements. If you insert or modify some rows and then commit that transaction, a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement issued from another concurrent **REPEATABLE READ** transaction could affect those just-committed rows, even though the session could not query them. If a transaction does update or delete rows committed by a different transaction, those changes do become visible to the current transaction. For example, you might encounter a situation like the following:

SELECT COUNT(c1) FROM t1 WHERE c1 = 'xyz';

-- Returns 0: no rows match.

DELETE FROM t1 WHERE c1 = 'xyz';

-- Deletes several rows recently committed by other transaction.

SELECT COUNT(c2) FROM t1 WHERE c2 = 'abc';

-- Returns 0: no rows match.

UPDATE t1 SET c2 = 'cba' WHERE c2 = 'abc';

-- Affects 10 rows: another txn just committed 10 rows with 'abc' values.

SELECT COUNT(c2) FROM t1 WHERE c2 = 'cba';

-- Returns 10: this txn can now see the rows it just updated.

You can advance your timepoint by committing your transaction and then doing another [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**START TRANSACTION WITH CONSISTENT SNAPSHOT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

This is called multi-versioned concurrency control.

In the following example, session A sees the row inserted by B only when B has committed the insert and A has committed as well, so that the timepoint is advanced past the commit of B.

Session A Session B

SET autocommit=0; SET autocommit=0;

time

| SELECT \* FROM t;

| empty set

| INSERT INTO t VALUES (1, 2);

|

v SELECT \* FROM t;

empty set

COMMIT;

SELECT \* FROM t;

empty set

COMMIT;

SELECT \* FROM t;

---------------------

| 1 | 2 |

---------------------

If you want to see the “freshest” state of the database, use either the [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level or a [locking read](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read):

SELECT \* FROM t FOR SHARE;

With [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level, each consistent read within a transaction sets and reads its own fresh snapshot. With **FOR SHARE**, a locking read occurs instead: A **SELECT** blocks until the transaction containing the freshest rows ends (see [Section 15.7.2.4, “Locking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads)).

Consistent read does not work over certain DDL statements:

Consistent read does not work over [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), because MySQL cannot use a table that has been dropped and **InnoDB** destroys the table.

Consistent read does not work over [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that make a temporary copy of the original table and delete the original table when the temporary copy is built. When you reissue a consistent read within a transaction, rows in the new table are not visible because those rows did not exist when the transaction's snapshot was taken. In this case, the transaction returns an error: [**ER\_TABLE\_DEF\_CHANGED**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_table_def_changed), “Table definition has changed, please retry transaction”.

The type of read varies for selects in clauses like [**INSERT INTO ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE ... (SELECT)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) that do not specify **FOR UPDATE** or **FOR SHARE**:

By default, **InnoDB** uses stronger locks for those statements and the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part acts like [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), where each consistent read, even within the same transaction, sets and reads its own fresh snapshot.

To perform a nonlocking read in such cases, set the isolation level of the transaction to [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted) or [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) to avoid setting locks on rows read from the selected table.

#### 15.7.2.4 Locking Reads

If you query data and then insert or update related data within the same transaction, the regular **SELECT** statement does not give enough protection. Other transactions can update or delete the same rows you just queried. **InnoDB** supports two types of [locking reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read) that offer extra safety:

[**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)

Sets a shared mode lock on any rows that are read. Other sessions can read the rows, but cannot modify them until your transaction commits. If any of these rows were changed by another transaction that has not yet committed, your query waits until that transaction ends and then uses the latest values.

**Note**

**SELECT ... FOR SHARE** is a replacement for **SELECT ... LOCK IN SHARE MODE**, but **LOCK IN SHARE MODE** remains available for backward compatibility. The statements are equivalent. However, **FOR SHARE** supports **OF *table\_name***, **NOWAIT**, and **SKIP LOCKED** options. See [Locking Read Concurrency with NOWAIT and SKIP LOCKED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads-nowait-skip-locked).

Prior to MySQL 8.0.22, **SELECT ... FOR SHARE** requires the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege and at least one of the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete), [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables), or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privileges. As of MySQL 8.0.22, only the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege is required.

As of MySQL 8.0.22, **SELECT ... FOR SHARE** statements do not acquire read locks on MySQL grant tables. For more information, see [Grant Table Concurrency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#grant-tables-concurrency).

[**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)

For index records the search encounters, locks the rows and any associated index entries, the same as if you issued an **UPDATE** statement for those rows. Other transactions are blocked from updating those rows, from doing **SELECT ... FOR SHARE**, or from reading the data in certain transaction isolation levels. Consistent reads ignore any locks set on the records that exist in the read view. (Old versions of a record cannot be locked; they are reconstructed by applying [undo logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_log) on an in-memory copy of the record.)

**SELECT ... FOR UPDATE** requires the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege and at least one of the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete), [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables), or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privileges.

These clauses are primarily useful when dealing with tree-structured or graph-structured data, either in a single table or split across multiple tables. You traverse edges or tree branches from one place to another, while reserving the right to come back and change any of these “pointer” values.

All locks set by **FOR SHARE** and **FOR UPDATE** queries are released when the transaction is committed or rolled back.

**Note**

Locking reads are only possible when autocommit is disabled (either by beginning transaction with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or by setting [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) to 0.

A locking read clause in an outer statement does not lock the rows of a table in a nested subquery unless a locking read clause is also specified in the subquery. For example, the following statement does not lock rows in table **t2**.

SELECT \* FROM t1 WHERE c1 = (SELECT c1 FROM t2) FOR UPDATE;

To lock rows in table **t2**, add a locking read clause to the subquery:

SELECT \* FROM t1 WHERE c1 = (SELECT c1 FROM t2 FOR UPDATE) FOR UPDATE;

##### Locking Read Examples

Suppose that you want to insert a new row into a table **child**, and make sure that the child row has a parent row in table **parent**. Your application code can ensure referential integrity throughout this sequence of operations.

First, use a consistent read to query the table **PARENT** and verify that the parent row exists. Can you safely insert the child row to table **CHILD**? No, because some other session could delete the parent row in the moment between your **SELECT** and your **INSERT**, without you being aware of it.

To avoid this potential issue, perform the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) using **FOR SHARE**:

SELECT \* FROM parent WHERE NAME = 'Jones' FOR SHARE;

After the **FOR SHARE** query returns the parent **'Jones'**, you can safely add the child record to the **CHILD** table and commit the transaction. Any transaction that tries to acquire an exclusive lock in the applicable row in the **PARENT** table waits until you are finished, that is, until the data in all tables is in a consistent state.

For another example, consider an integer counter field in a table **CHILD\_CODES**, used to assign a unique identifier to each child added to table **CHILD**. Do not use either consistent read or a shared mode read to read the present value of the counter, because two users of the database could see the same value for the counter, and a duplicate-key error occurs if two transactions attempt to add rows with the same identifier to the **CHILD** table.

Here, **FOR SHARE** is not a good solution because if two users read the counter at the same time, at least one of them ends up in deadlock when it attempts to update the counter.

To implement reading and incrementing the counter, first perform a locking read of the counter using **FOR UPDATE**, and then increment the counter. For example:

SELECT counter\_field FROM child\_codes FOR UPDATE;

UPDATE child\_codes SET counter\_field = counter\_field + 1;

A [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) reads the latest available data, setting exclusive locks on each row it reads. Thus, it sets the same locks a searched SQL [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) would set on the rows.

The preceding description is merely an example of how [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) works. In MySQL, the specific task of generating a unique identifier actually can be accomplished using only a single access to the table:

UPDATE child\_codes SET counter\_field = LAST\_INSERT\_ID(counter\_field + 1);

SELECT LAST\_INSERT\_ID();

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement merely retrieves the identifier information (specific to the current connection). It does not access any table.

##### Locking Read Concurrency with NOWAIT and SKIP LOCKED

If a row is locked by a transaction, a **SELECT ... FOR UPDATE** or **SELECT ... FOR SHARE** transaction that requests the same locked row must wait until the blocking transaction releases the row lock. This behavior prevents transactions from updating or deleting rows that are queried for updates by other transactions. However, waiting for a row lock to be released is not necessary if you want the query to return immediately when a requested row is locked, or if excluding locked rows from the result set is acceptable.

To avoid waiting for other transactions to release row locks, **NOWAIT** and **SKIP LOCKED** options may be used with **SELECT ... FOR UPDATE** or **SELECT ... FOR SHARE** locking read statements.

**NOWAIT**

A locking read that uses **NOWAIT** never waits to acquire a row lock. The query executes immediately, failing with an error if a requested row is locked.

**SKIP LOCKED**

A locking read that uses **SKIP LOCKED** never waits to acquire a row lock. The query executes immediately, removing locked rows from the result set.

**Note**

Queries that skip locked rows return an inconsistent view of the data. **SKIP LOCKED** is therefore not suitable for general transactional work. However, it may be used to avoid lock contention when multiple sessions access the same queue-like table.

**NOWAIT** and **SKIP LOCKED** only apply to row-level locks.

Statements that use **NOWAIT** or **SKIP LOCKED** are unsafe for statement based replication.

The following example demonstrates **NOWAIT** and **SKIP LOCKED**. Session 1 starts a transaction that takes a row lock on a single record. Session 2 attempts a locking read on the same record using the **NOWAIT** option. Because the requested row is locked by Session 1, the locking read returns immediately with an error. In Session 3, the locking read with **SKIP LOCKED** returns the requested rows except for the row that is locked by Session 1.

# Session 1:

mysql> **CREATE TABLE t (i INT, PRIMARY KEY (i)) ENGINE = InnoDB;**

mysql> **INSERT INTO t (i) VALUES(1),(2),(3);**

mysql> **START TRANSACTION;**

mysql> **SELECT \* FROM t WHERE i = 2 FOR UPDATE;**

+---+

| i |

+---+

| 2 |

+---+

# Session 2:

mysql> **START TRANSACTION;**

mysql> **SELECT \* FROM t WHERE i = 2 FOR UPDATE NOWAIT;**

ERROR 3572 (HY000): Do not wait for lock.

# Session 3:

mysql> **START TRANSACTION;**

mysql> **SELECT \* FROM t FOR UPDATE SKIP LOCKED;**

+---+

| i |

+---+

| 1 |

| 3 |

+---+

### 15.7.3 Locks Set by Different SQL Statements in InnoDB

A [locking read](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read), an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) generally set record locks on every index record that is scanned in the processing of the SQL statement. It does not matter whether there are **WHERE** conditions in the statement that would exclude the row. **InnoDB** does not remember the exact **WHERE** condition, but only knows which index ranges were scanned. The locks are normally [next-key locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_next_key_lock) that also block inserts into the “gap” immediately before the record. However, [gap locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap_lock) can be disabled explicitly, which causes next-key locking not to be used. For more information, see [Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking). The transaction isolation level also can affect which locks are set; see [Section 15.7.2.1, “Transaction Isolation Levels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-isolation-levels).

If a secondary index is used in a search and index record locks to be set are exclusive, **InnoDB** also retrieves the corresponding clustered index records and sets locks on them.

If you have no indexes suitable for your statement and MySQL must scan the entire table to process the statement, every row of the table becomes locked, which in turn blocks all inserts by other users to the table. It is important to create good indexes so that your queries do not unnecessarily scan many rows.

**InnoDB** sets specific types of locks as follows.

[**SELECT ... FROM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is a consistent read, reading a snapshot of the database and setting no locks unless the transaction isolation level is set to [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable). For [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) level, the search sets shared next-key locks on the index records it encounters. However, only an index record lock is required for statements that lock rows using a unique index to search for a unique row.

[**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements that use a unique index acquire locks for scanned rows, and release the locks for rows that do not qualify for inclusion in the result set (for example, if they do not meet the criteria given in the **WHERE** clause). However, in some cases, rows might not be unlocked immediately because the relationship between a result row and its original source is lost during query execution. For example, in a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), scanned (and locked) rows from a table might be inserted into a temporary table before evaluation whether they qualify for the result set. In this circumstance, the relationship of the rows in the temporary table to the rows in the original table is lost and the latter rows are not unlocked until the end of query execution.

For [locking reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read) ([**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) with **FOR UPDATE** or **FOR SHARE**), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements, the locks that are taken depend on whether the statement uses a unique index with a unique search condition, or a range-type search condition.

For a unique index with a unique search condition, **InnoDB** locks only the index record found, not the [gap](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap) before it.

For other search conditions, and for non-unique indexes, **InnoDB** locks the index range scanned, using [gap locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_gap_lock) or [next-key locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_next_key_lock) to block insertions by other sessions into the gaps covered by the range. For information about gap locks and next-key locks, see [Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking).

For index records the search encounters, [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) blocks other sessions from doing [**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or from reading in certain transaction isolation levels. Consistent reads ignore any locks set on the records that exist in the read view.

[**UPDATE ... WHERE ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) sets an exclusive next-key lock on every record the search encounters. However, only an index record lock is required for statements that lock rows using a unique index to search for a unique row.

When [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) modifies a clustered index record, implicit locks are taken on affected secondary index records. The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operation also takes shared locks on affected secondary index records when performing duplicate check scans prior to inserting new secondary index records, and when inserting new secondary index records.

[**DELETE FROM ... WHERE ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) sets an exclusive next-key lock on every record the search encounters. However, only an index record lock is required for statements that lock rows using a unique index to search for a unique row.

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) sets an exclusive lock on the inserted row. This lock is an index-record lock, not a next-key lock (that is, there is no gap lock) and does not prevent other sessions from inserting into the gap before the inserted row.

Prior to inserting the row, a type of gap lock called an insert intention gap lock is set. This lock signals the intent to insert in such a way that multiple transactions inserting into the same index gap need not wait for each other if they are not inserting at the same position within the gap. Suppose that there are index records with values of 4 and 7. Separate transactions that attempt to insert values of 5 and 6 each lock the gap between 4 and 7 with insert intention locks prior to obtaining the exclusive lock on the inserted row, but do not block each other because the rows are nonconflicting.

If a duplicate-key error occurs, a shared lock on the duplicate index record is set. This use of a shared lock can result in deadlock should there be multiple sessions trying to insert the same row if another session already has an exclusive lock. This can occur if another session deletes the row. Suppose that an **InnoDB** table **t1** has the following structure:

CREATE TABLE t1 (i INT, PRIMARY KEY (i)) ENGINE = InnoDB;

Now suppose that three sessions perform the following operations in order:

Session 1:

START TRANSACTION;

INSERT INTO t1 VALUES(1);

Session 2:

START TRANSACTION;

INSERT INTO t1 VALUES(1);

Session 3:

START TRANSACTION;

INSERT INTO t1 VALUES(1);

Session 1:

ROLLBACK;

The first operation by session 1 acquires an exclusive lock for the row. The operations by sessions 2 and 3 both result in a duplicate-key error and they both request a shared lock for the row. When session 1 rolls back, it releases its exclusive lock on the row and the queued shared lock requests for sessions 2 and 3 are granted. At this point, sessions 2 and 3 deadlock: Neither can acquire an exclusive lock for the row because of the shared lock held by the other.

A similar situation occurs if the table already contains a row with key value 1 and three sessions perform the following operations in order:

Session 1:

START TRANSACTION;

DELETE FROM t1 WHERE i = 1;

Session 2:

START TRANSACTION;

INSERT INTO t1 VALUES(1);

Session 3:

START TRANSACTION;

INSERT INTO t1 VALUES(1);

Session 1:

COMMIT;

The first operation by session 1 acquires an exclusive lock for the row. The operations by sessions 2 and 3 both result in a duplicate-key error and they both request a shared lock for the row. When session 1 commits, it releases its exclusive lock on the row and the queued shared lock requests for sessions 2 and 3 are granted. At this point, sessions 2 and 3 deadlock: Neither can acquire an exclusive lock for the row because of the shared lock held by the other.

[**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) differs from a simple [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) in that an exclusive lock rather than a shared lock is placed on the row to be updated when a duplicate-key error occurs. An exclusive index-record lock is taken for a duplicate primary key value. An exclusive next-key lock is taken for a duplicate unique key value.

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) is done like an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) if there is no collision on a unique key. Otherwise, an exclusive next-key lock is placed on the row to be replaced.

**INSERT INTO T SELECT ... FROM S WHERE ...** sets an exclusive index record lock (without a gap lock) on each row inserted into **T**. If the transaction isolation level is [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), **InnoDB** does the search on **S** as a consistent read (no locks). Otherwise, **InnoDB** sets shared next-key locks on rows from **S**. **InnoDB** has to set locks in the latter case: During roll-forward recovery using a statement-based binary log, every SQL statement must be executed in exactly the same way it was done originally.

[**CREATE TABLE ... SELECT ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) performs the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) with shared next-key locks or as a consistent read, as for [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select).

When a **SELECT** is used in the constructs **REPLACE INTO t SELECT ... FROM s WHERE ...** or **UPDATE t ... WHERE col IN (SELECT ... FROM s ...)**, **InnoDB** sets shared next-key locks on rows from table **s**.

**InnoDB** sets an exclusive lock on the end of the index associated with the **AUTO\_INCREMENT** column while initializing a previously specified **AUTO\_INCREMENT** column on a table.

With [**innodb\_autoinc\_lock\_mode=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode), **InnoDB** uses a special **AUTO-INC** table lock mode where the lock is obtained and held to the end of the current SQL statement (not to the end of the entire transaction) while accessing the auto-increment counter. Other clients cannot insert into the table while the **AUTO-INC** table lock is held. The same behavior occurs for “bulk inserts” with [**innodb\_autoinc\_lock\_mode=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode). Table-level **AUTO-INC** locks are not used with [**innodb\_autoinc\_lock\_mode=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode). For more information, See [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling).

**InnoDB** fetches the value of a previously initialized **AUTO\_INCREMENT** column without setting any locks.

If a **FOREIGN KEY** constraint is defined on a table, any insert, update, or delete that requires the constraint condition to be checked sets shared record-level locks on the records that it looks at to check the constraint. **InnoDB** also sets these locks in the case where the constraint fails.

[**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) sets table locks, but it is the higher MySQL layer above the **InnoDB** layer that sets these locks. **InnoDB** is aware of table locks if **innodb\_table\_locks = 1** (the default) and [**autocommit = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), and the MySQL layer above **InnoDB** knows about row-level locks.

Otherwise, **InnoDB**'s automatic deadlock detection cannot detect deadlocks where such table locks are involved. Also, because in this case the higher MySQL layer does not know about row-level locks, it is possible to get a table lock on a table where another session currently has row-level locks. However, this does not endanger transaction integrity, as discussed in [Section 15.7.5.2, “Deadlock Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection).

[**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) acquires two locks on each table if **innodb\_table\_locks=1** (the default). In addition to a table lock on the MySQL layer, it also acquires an **InnoDB** table lock. Versions of MySQL before 4.1.2 did not acquire **InnoDB** table locks; the old behavior can be selected by setting **innodb\_table\_locks=0**. If no **InnoDB** table lock is acquired, [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) completes even if some records of the tables are being locked by other transactions.

In MySQL 8.0, [**innodb\_table\_locks=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_table_locks) has no effect for tables locked explicitly with [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). It does have an effect for tables locked for read or write by [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) implicitly (for example, through triggers) or by [**LOCK TABLES ... READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

All **InnoDB** locks held by a transaction are released when the transaction is committed or aborted. Thus, it does not make much sense to invoke [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) on **InnoDB** tables in [**autocommit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode because the acquired **InnoDB** table locks would be released immediately.

You cannot lock additional tables in the middle of a transaction because [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) performs an implicit [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

### 15.7.4 Phantom Rows

The so-called phantom problem occurs within a transaction when the same query produces different sets of rows at different times. For example, if a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is executed twice, but returns a row the second time that was not returned the first time, the row is a “phantom” row.

Suppose that there is an index on the **id** column of the **child** table and that you want to read and lock all rows from the table having an identifier value larger than 100, with the intention of updating some column in the selected rows later:

SELECT \* FROM child WHERE id > 100 FOR UPDATE;

The query scans the index starting from the first record where **id** is bigger than 100. Let the table contain rows having **id** values of 90 and 102. If the locks set on the index records in the scanned range do not lock out inserts made in the gaps (in this case, the gap between 90 and 102), another session can insert a new row into the table with an **id** of 101. If you were to execute the same [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) within the same transaction, you would see a new row with an **id** of 101 (a “phantom”) in the result set returned by the query. If we regard a set of rows as a data item, the new phantom child would violate the isolation principle of transactions that a transaction should be able to run so that the data it has read does not change during the transaction.

To prevent phantoms, **InnoDB** uses an algorithm called next-key locking that combines index-row locking with gap locking. **InnoDB** performs row-level locking in such a way that when it searches or scans a table index, it sets shared or exclusive locks on the index records it encounters. Thus, the row-level locks are actually index-record locks. In addition, a next-key lock on an index record also affects the “gap” before that index record. That is, a next-key lock is an index-record lock plus a gap lock on the gap preceding the index record. If one session has a shared or exclusive lock on record **R** in an index, another session cannot insert a new index record in the gap immediately before **R** in the index order.

When **InnoDB** scans an index, it can also lock the gap after the last record in the index. Just that happens in the preceding example: To prevent any insert into the table where **id** would be bigger than 100, the locks set by **InnoDB** include a lock on the gap following **id** value 102.

You can use next-key locking to implement a uniqueness check in your application: If you read your data in share mode and do not see a duplicate for a row you are going to insert, then you can safely insert your row and know that the next-key lock set on the successor of your row during the read prevents anyone meanwhile inserting a duplicate for your row. Thus, the next-key locking enables you to “lock” the nonexistence of something in your table.

Gap locking can be disabled as discussed in [Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking). This may cause phantom problems because other sessions can insert new rows into the gaps when gap locking is disabled.

### 15.7.5 Deadlocks in InnoDB

[15.7.5.1 An InnoDB Deadlock Example](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-example)

[15.7.5.2 Deadlock Detection](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection)

[15.7.5.3 How to Minimize and Handle Deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks-handling)

A deadlock is a situation where different transactions are unable to proceed because each holds a lock that the other needs. Because both transactions are waiting for a resource to become available, neither ever release the locks it holds.

A deadlock can occur when transactions lock rows in multiple tables (through statements such as [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)), but in the opposite order. A deadlock can also occur when such statements lock ranges of index records and gaps, with each transaction acquiring some locks but not others due to a timing issue. For a deadlock example, see [Section 15.7.5.1, “An InnoDB Deadlock Example”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-example).

To reduce the possibility of deadlocks, use transactions rather than [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statements; keep transactions that insert or update data small enough that they do not stay open for long periods of time; when different transactions update multiple tables or large ranges of rows, use the same order of operations (such as [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)) in each transaction; create indexes on the columns used in [**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**UPDATE ... WHERE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements. The possibility of deadlocks is not affected by the isolation level, because the isolation level changes the behavior of read operations, while deadlocks occur because of write operations. For more information about avoiding and recovering from deadlock conditions, see [Section 15.7.5.3, “How to Minimize and Handle Deadlocks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks-handling).

When deadlock detection is enabled (the default) and a deadlock does occur, **InnoDB** detects the condition and rolls back one of the transactions (the victim). If deadlock detection is disabled using the [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) configuration option, **InnoDB** relies on the [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) setting to roll back transactions in case of a deadlock. Thus, even if your application logic is correct, you must still handle the case where a transaction must be retried. To see the last deadlock in an **InnoDB** user transaction, use the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) command. If frequent deadlocks highlight a problem with transaction structure or application error handling, run with the [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) setting enabled to print information about all deadlocks to the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) error log. For more information about how deadlocks are automatically detected and handled, see [Section 15.7.5.2, “Deadlock Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection).

#### 15.7.5.1 An InnoDB Deadlock Example

The following example illustrates how an error can occur when a lock request would cause a deadlock. The example involves two clients, A and B.

First, client A creates a table containing one row, and then begins a transaction. Within the transaction, A obtains an **S** lock on the row by selecting it in share mode:

mysql> **CREATE TABLE t (i INT) ENGINE = InnoDB;**

Query OK, 0 rows affected (1.07 sec)

mysql> **INSERT INTO t (i) VALUES(1);**

Query OK, 1 row affected (0.09 sec)

mysql> **START TRANSACTION;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT \* FROM t WHERE i = 1 FOR SHARE;**

+------+

| i |

+------+

| 1 |

+------+

Next, client B begins a transaction and attempts to delete the row from the table:

mysql> **START TRANSACTION;**

Query OK, 0 rows affected (0.00 sec)

mysql> **DELETE FROM t WHERE i = 1;**

The delete operation requires an **X** lock. The lock cannot be granted because it is incompatible with the **S** lock that client A holds, so the request goes on the queue of lock requests for the row and client B blocks.

Finally, client A also attempts to delete the row from the table:

mysql> **DELETE FROM t WHERE i = 1;**

ERROR 1213 (40001): Deadlock found when trying to get lock;

try restarting transaction

Deadlock occurs here because client A needs an **X** lock to delete the row. However, that lock request cannot be granted because client B already has a request for an **X** lock and is waiting for client A to release its **S** lock. Nor can the **S** lock held by A be upgraded to an **X** lock because of the prior request by B for an **X** lock. As a result, **InnoDB** generates an error for one of the clients and releases its locks. The client returns this error:

ERROR 1213 (40001): Deadlock found when trying to get lock;

try restarting transaction

At that point, the lock request for the other client can be granted and it deletes the row from the table.

#### 15.7.5.2 Deadlock Detection

When [deadlock detection](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock_detection) is enabled (the default), **InnoDB** automatically detects transaction [deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) and rolls back a transaction or transactions to break the deadlock. **InnoDB** tries to pick small transactions to roll back, where the size of a transaction is determined by the number of rows inserted, updated, or deleted.

**InnoDB** is aware of table locks if **innodb\_table\_locks = 1** (the default) and [**autocommit = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), and the MySQL layer above it knows about row-level locks. Otherwise, **InnoDB** cannot detect deadlocks where a table lock set by a MySQL [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement or a lock set by a storage engine other than **InnoDB** is involved. Resolve these situations by setting the value of the [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) system variable.

If the **LATEST DETECTED DEADLOCK** section of **InnoDB** Monitor output includes a message stating TOO DEEP OR LONG SEARCH IN THE LOCK TABLE WAITS-FOR GRAPH, WE WILL ROLL BACK FOLLOWING TRANSACTION, this indicates that the number of transactions on the wait-for list has reached a limit of 200. A wait-for list that exceeds 200 transactions is treated as a deadlock and the transaction attempting to check the wait-for list is rolled back. The same error may also occur if the locking thread must look at more than 1,000,000 locks owned by transactions on the wait-for list.

For techniques to organize database operations to avoid deadlocks, see [Section 15.7.5, “Deadlocks in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks).

##### Disabling Deadlock Detection

On high concurrency systems, deadlock detection can cause a slowdown when numerous threads wait for the same lock. At times, it may be more efficient to disable deadlock detection and rely on the [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) setting for transaction rollback when a deadlock occurs. Deadlock detection can be disabled using the [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) configuration option.

#### 15.7.5.3 How to Minimize and Handle Deadlocks

This section builds on the conceptual information about deadlocks in [Section 15.7.5.2, “Deadlock Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection). It explains how to organize database operations to minimize deadlocks and the subsequent error handling required in applications.

[Deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) are a classic problem in transactional databases, but they are not dangerous unless they are so frequent that you cannot run certain transactions at all. Normally, you must write your applications so that they are always prepared to re-issue a transaction if it gets rolled back because of a deadlock.

**InnoDB** uses automatic row-level locking. You can get deadlocks even in the case of transactions that just insert or delete a single row. That is because these operations are not really “atomic”; they automatically set locks on the (possibly several) index records of the row inserted or deleted.

You can cope with deadlocks and reduce the likelihood of their occurrence with the following techniques:

At any time, issue the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) command to determine the cause of the most recent deadlock. That can help you to tune your application to avoid deadlocks.

If frequent deadlock warnings cause concern, collect more extensive debugging information by enabling the [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) configuration option. Information about each deadlock, not just the latest one, is recorded in the MySQL [error log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_error_log). Disable this option when you are finished debugging.

Always be prepared to re-issue a transaction if it fails due to deadlock. Deadlocks are not dangerous. Just try again.

Keep transactions small and short in duration to make them less prone to collision.

Commit transactions immediately after making a set of related changes to make them less prone to collision. In particular, do not leave an interactive [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) session open for a long time with an uncommitted transaction.

If you use [locking reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking_read) ([**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or **SELECT ... FOR SHARE**), try using a lower isolation level such as [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed).

When modifying multiple tables within a transaction, or different sets of rows in the same table, do those operations in a consistent order each time. Then transactions form well-defined queues and do not deadlock. For example, organize database operations into functions within your application, or call stored routines, rather than coding multiple similar sequences of **INSERT**, **UPDATE**, and **DELETE** statements in different places.

Add well-chosen indexes to your tables. Then your queries need to scan fewer index records and consequently set fewer locks. Use [**EXPLAIN SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) to determine which indexes the MySQL server regards as the most appropriate for your queries.

Use less locking. If you can afford to permit a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) to return data from an old snapshot, do not add the clause **FOR UPDATE** or **FOR SHARE** to it. Using the [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed) isolation level is good here, because each consistent read within the same transaction reads from its own fresh snapshot.

If nothing else helps, serialize your transactions with table-level locks. The correct way to use [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) with transactional tables, such as **InnoDB** tables, is to begin a transaction with **SET autocommit = 0** (not [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit)) followed by [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), and to not call [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) until you commit the transaction explicitly. For example, if you need to write to table **t1** and read from table **t2**, you can do this:

SET autocommit=0;

LOCK TABLES t1 WRITE, t2 READ, ...;

***... do something with tables t1 and t2 here ...***

COMMIT;

UNLOCK TABLES;

Table-level locks prevent concurrent updates to the table, avoiding deadlocks at the expense of less responsiveness for a busy system.

Another way to serialize transactions is to create an auxiliary “semaphore” table that contains just a single row. Have each transaction update that row before accessing other tables. In that way, all transactions happen in a serial fashion. Note that the **InnoDB** instant deadlock detection algorithm also works in this case, because the serializing lock is a row-level lock. With MySQL table-level locks, the timeout method must be used to resolve deadlocks.

### 15.7.6 Transaction Scheduling

**InnoDB** uses the Contention-Aware Transaction Scheduling (CATS) algorithm to prioritize transactions that are waiting for locks. When multiple transactions are waiting for a lock on the same object, the CATS algorithm determines which transaction receives the lock first.

The CATS algorithm prioritizes waiting transactions by assigning a scheduling weight, which is computed based on the number of transactions that a transaction blocks. For example, if two transactions are waiting for a lock on the same object, the transaction that blocks the most transactions is assigned a greater scheduling weight. If weights are equal, priority is given to the longest waiting transaction.

**Note**

Prior to MySQL 8.0.20, **InnoDB** also uses a First In First Out (FIFO) algorithm to schedule transactions, and the CATS algorithm is used under heavy lock contention only. CATS algorithm enhancements in MySQL 8.0.20 rendered the FIFO algorithm redundant, permitting its removal. Transaction scheduling previously performed by the FIFO algorithm is performed by the CATS algorithm as of MySQL 8.0.20. In some cases, this change may affect the order in which transactions are granted locks.

You can view transaction scheduling weights by querying the **TRX\_SCHEDULE\_WEIGHT** column in the [**INFORMATION\_SCHEMA.INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table. Weights are computed for waiting transactions only. Waiting transactions are those in a **LOCK WAIT** transaction execution state, as reported by the **TRX\_STATE** column. A transaction that is not waiting for a lock reports a NULL **TRX\_SCHEDULE\_WEIGHT** value.

[**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) counters are provided for monitoring of code-level transaction scheduling events. For information about using [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) counters, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

**lock\_rec\_release\_attempts**

The number of attempts to release record locks. A single attempt may lead to zero or more record locks being released, as there may be zero or more record locks in a single structure.

**lock\_rec\_grant\_attempts**

The number of attempts to grant record locks. A single attempt may result in zero or more record locks being granted.

**lock\_schedule\_refreshes**

The number of times the wait-for graph was analyzed to update the scheduled transaction weights.

## 15.8 InnoDB Configuration

[15.8.1 InnoDB Startup Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-init-startup-configuration)

[15.8.2 Configuring InnoDB for Read-Only Operation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-read-only-instance)

[15.8.3 InnoDB Buffer Pool Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-buffer-pool)

[15.8.4 Configuring Thread Concurrency for InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-thread_concurrency)

[15.8.5 Configuring the Number of Background InnoDB I/O Threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-multiple_io_threads)

[15.8.6 Using Asynchronous I/O on Linux](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-linux-native-aio)

[15.8.7 Configuring InnoDB I/O Capacity](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-configuring-io-capacity)

[15.8.8 Configuring Spin Lock Polling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-spin_lock_polling)

[15.8.9 Purge Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration)

[15.8.10 Configuring Optimizer Statistics for InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-optimizer-statistics)

[15.8.11 Configuring the Merge Threshold for Index Pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold)

[15.8.12 Enabling Automatic Configuration for a Dedicated MySQL Server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-dedicated-server)

This section provides configuration information and procedures for **InnoDB** initialization, startup, and various components and features of the **InnoDB** storage engine. For information about optimizing database operations for **InnoDB** tables, see [Section 8.5, “Optimizing for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb).

### 15.8.1 InnoDB Startup Configuration

The first decisions to make about **InnoDB** configuration involve the configuration of data files, log files, page size, and memory buffers. It is recommended that you define data file, log file, and page size configuration before creating the **InnoDB** instance. Modifying data file or log file configuration after the **InnoDB** instance is created may involve a non-trivial procedure, and page size can only be defined when the **InnoDB** instance is first initialized.

In addition to these topics, this section provides information about specifying **InnoDB** options in a configuration file, viewing **InnoDB** initialization information, and important storage considerations.

[Specifying Options in a MySQL Configuration File](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-mysql-configuration-file)

[Viewing InnoDB Initialization Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-initialization-information)

[Important Storage Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-storage-considerations)

[System Tablespace Data File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-data-file-configuration)

[InnoDB Doublewrite Buffer File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-doublewrite-file-config)

[Redo Log File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-log-file-configuration)

[Undo Tablespace Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-undo-tablespace)

[Global Temporary Tablespace Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-temporary-tablespace)

[Session Temporary Tablespace Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-session-temporary-tablespaces)

[Page Size Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-page-size)

[Memory Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-memory-configuration)

#### Specifying Options in a MySQL Configuration File

Because MySQL uses data file, log file, and page size configuration settings to initialize the **InnoDB** instance, it is recommended that you define these settings in a configuration file that MySQL reads at startup, prior to initializing **InnoDB** for the first time. **InnoDB** is initialized when the MySQL server is started, and the first initialization of **InnoDB** normally occurs the first time you start the MySQL server.

You can place **InnoDB** options in the **[mysqld]** group of any option file that your server reads when it starts. The locations of MySQL option files are described in [Section 4.2.2.2, “Using Option Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files).

To make sure that [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) reads options only from a specific file (and mysqld-auto.cnf), use the [--defaults-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_defaults-file) option as the first option on the command line when starting the server:

mysqld --defaults-file=***path\_to\_configuration\_file***

#### Viewing InnoDB Initialization Information

To view **InnoDB** initialization information during startup, start [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) from a command prompt. When [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started from a command prompt, initialization information is printed to the console.

For example, on Windows, if [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is located in C:\Program Files\MySQL\MySQL Server 8.0\bin, start the MySQL server like this:

C:\> **"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console**

On Unix-like systems, [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is located in the bin directory of your MySQL installation:

shell> **bin/mysqld --user=mysql &**

If you do not send server output to the console, check the error log after startup to see the initialization information **InnoDB** printed during the startup process.

For information about starting MySQL using other methods, see [Section 2.10.5, “Starting and Stopping MySQL Automatically”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#automatic-start).

**Note**

**InnoDB** does not open all user tables and associated data files at startup. However, **InnoDB** does check for the existence of tablespace files (\*.ibd files) that are referenced in the data dictionary. If a tablespace file is not found, **InnoDB** logs an error and continues the startup sequence. Tablespace files that are referenced in the redo log may be opened during crash recovery for redo application.

#### Important Storage Considerations

Review the following storage-related considerations before proceeding with your startup configuration.

In some cases, database performance improves if the data is not all placed on the same physical disk. Putting log files on a different disk from data is very often beneficial for performance. For example, you can place system tablespace data files and log files on different disks. You can also use raw disk partitions (raw devices) for **InnoDB** data files, which may speed up I/O. See [Using Raw Disk Partitions for the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-raw-devices).

**InnoDB** is a transaction-safe (ACID compliant) storage engine for MySQL that has commit, rollback, and crash-recovery capabilities to protect user data. ***However, it cannot do so*** if the underlying operating system or hardware does not work as advertised. Many operating systems or disk subsystems may delay or reorder write operations to improve performance. On some operating systems, the very **fsync()** system call that should wait until all unwritten data for a file has been flushed might actually return before the data has been flushed to stable storage. Because of this, an operating system crash or a power outage may destroy recently committed data, or in the worst case, even corrupt the database because of write operations having been reordered. If data integrity is important to you, perform some “pull-the-plug” tests before using anything in production. On macOS, **InnoDB** uses a special **fcntl()** file flush method. Under Linux, it is advisable to ***disable the write-back cache***.

On ATA/SATA disk drives, a command such **hdparm -W0 /dev/hda** may work to disable the write-back cache. ***Beware that some drives or disk controllers may be unable to disable the write-back cache.***

With regard to **InnoDB** recovery capabilities that protect user data, **InnoDB** uses a file flush technique involving a structure called the [doublewrite buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_doublewrite_buffer), which is enabled by default ([**innodb\_doublewrite=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite)). The doublewrite buffer adds safety to recovery following an unexpected exit or power outage, and improves performance on most varieties of Unix by reducing the need for **fsync()** operations. It is recommended that the [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) option remains enabled if you are concerned with data integrity or possible failures. For additional information about the doublewrite buffer, see [Section 15.11.1, “InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disk-io).

Before using NFS with **InnoDB**, review potential issues outlined in [Using NFS with MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#disk-issues-nfs).

#### System Tablespace Data File Configuration

The [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) startup option defines the name, size, and attributes of **InnoDB** system tablespace data files. If you do not configure this option prior to initializing the MySQL server, the default behavior is to create a single auto-extending data file, slightly larger than 12MB, named ibdata1:

mysql> **SHOW VARIABLES LIKE 'innodb\_data\_file\_path';**

+-----------------------+------------------------+

| Variable\_name | Value |

+-----------------------+------------------------+

| innodb\_data\_file\_path | ibdata1:12M:autoextend |

+-----------------------+------------------------+

The full data file specification syntax includes the file name, file size, **autoextend** attribute, and **max** attribute:

***file\_name***:***file\_size***[:autoextend[:max:***max\_file\_size***]]

File sizes are specified in kilobytes, megabytes, or gigabytes by appending **K**, **M** or **G** to the size value. If specifying the data file size in kilobytes, do so in multiples of 1024. Otherwise, kilobyte values are rounded to nearest megabyte (MB) boundary. The sum of file sizes must be, at a minimum, slightly larger than 12MB.

You can specify more than one data file using a semicolon-separated list. For example:

[mysqld]

innodb\_data\_file\_path=ibdata1:50M;ibdata2:50M:autoextend

The **autoextend** and **max** attributes can be used only for the data file that is specified last.

When the **autoextend** attribute is specified, the data file automatically increases in size by 64MB increments as space is required. The [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) variable controls the increment size.

To specify a maximum size for an auto-extending data file, use the **max** attribute following the **autoextend** attribute. Use the **max** attribute only in cases where constraining disk usage is of critical importance. The following configuration permits ibdata1 to grow to a limit of 500MB:

[mysqld]

innodb\_data\_file\_path=ibdata1:12M:autoextend:max:500M

A minimum file size is enforced for the first system tablespace data file to ensure that there is enough space for doublewrite buffer pages. The following table shows minimum file sizes for each **InnoDB** page size. The default **InnoDB** page size is 16384 (16KB).

| **Page Size (innodb\_page\_size)** | **Minimum File Size** |
| --- | --- |
| 16384 (16KB) or less | 3MB |
| 32768 (32KB) | 6MB |
| 65536 (64KB) | 12MB |

If your disk becomes full, you can add a data file on another disk. For instructions, see [Resizing the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-resize-system-tablespace).

The size limit for individual files is determined by your operating system. You can set the file size to more than 4GB on operating systems that support large files. You can also use raw disk partitions as data files. See [Using Raw Disk Partitions for the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-raw-devices).

**InnoDB** is not aware of the file system maximum file size, so be cautious on file systems where the maximum file size is a small value such as 2GB.

System tablespace files are created in the data directory by default ([**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir)). To specify an alternate location, you can use the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) option. For example, to create a system tablespace data file in a directory named myibdata, use this configuration:

[mysqld]

innodb\_data\_home\_dir = /myibdata/

innodb\_data\_file\_path=ibdata1:50M:autoextend

A trailing slash is required when specifying a value for [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir). **InnoDB** does not create directories, so ensure that the specified directory exists before you start the server. Also, ensure sure that the MySQL server has the proper access rights to create files in the directory.

**InnoDB** forms the directory path for each data file by textually concatenating the value of [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) to the data file name. If [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) is not defined, the default value is “./”, which is the data directory. (The MySQL server changes its current working directory to the data directory when it begins executing.)

Alternatively, you can specify an absolute path for system tablespace data files. The following configuration is equivalent to the preceding one:

[mysqld]

innodb\_data\_file\_path=/myibdata/ibdata1:50M:autoextend

When you specify an absolute path for [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path), the setting is not concatenated with the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) setting. System tablespace files are created in the specified absolute path. The specified directory must exist before you start the server.

#### InnoDB Doublewrite Buffer File Configuration

As of MySQL 8.0.20, the doublewrite buffer storage area resides in doublewrite files, which provides flexibility with respect to the storage location of doublewrite pages. In previous releases, the doublwrite buffer storage area resided in the system tablespace. The [**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir) variable defines the directory where **InnoDB** creates doublewrite files at startup. If no directory is specified, doublewrite files are created in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory, which defaults to the data directory if unspecified.

To have doublewrite files created in a location other than the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory, configure [**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir) variable. For example:

innodb\_doublewrite\_dir=***/path/to/doublewrite\_directory***

Other doublewrite buffer variables permit defining the number of doublewrite files, the number of pages per thread, and the doublewrite batch size. For more information about doublewrite buffer configuration, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

#### Redo Log File Configuration

By default, **InnoDB** creates two 5MB redo log files in the data directory named ib\_logfile0 and ib\_logfile1.

The following options can be used to modify the default configuration:

[**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir) defines directory path to the **InnoDB** log files (the redo logs). If this option is not configured, **InnoDB** log files are created in the MySQL data directory ([**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir)).

You might use this option to place **InnoDB** log files in a different physical storage location than **InnoDB** data files to avoid potential I/O resource conflicts. For example:

[mysqld]

innodb\_log\_group\_home\_dir = /dr3/iblogs

**Note**

**InnoDB** does not create directories, so make sure that the log directory exists before you start the server. Use the Unix or DOS **mkdir** command to create any necessary directories.

Make sure that the MySQL server has the proper access rights to create files in the log directory. More generally, the server must have access rights in any directory where it needs to create log files.

[**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) defines the number of log files in the log group. The default and recommended value is 2.

[**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) defines the size in bytes of each log file in the log group. The combined size of log files ([**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) \* [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group)) cannot exceed a maximum value that is slightly less than 512GB. A pair of 255 GB log files, for example, approaches the limit but does not exceed it. The default log file size is 48MB. Generally, the combined size of the log files should be large enough that the server can smooth out peaks and troughs in workload activity, which often means that there is enough redo log space to handle more than an hour of write activity. The larger the value, the less checkpoint flush activity is needed in the buffer pool, saving disk I/O. For additional information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

#### Undo Tablespace Configuration

By default, undo logs reside in two undo tablespaces that are created when the MySQL instance is initialized. The I/O patterns for undo logs make undo tablespaces good candidates for [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) storage.

The [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable defines the path where **InnoDB** creates default undo tablespaces. If that variable is undefined, default undo tablespaces are created in the data directory. The [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is not dynamic. Configuring it requires restarting the server.

For information about configuring additional undo tablespaces, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

#### Global Temporary Tablespace Configuration

The global temporary tablespace stores rollback segments for changes made to user-created temporary tables.

By default, **InnoDB** creates a single auto-extending global temporary tablespace data file named ibtmp1 in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory. The initial file size is slightly larger than 12MB.

The [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) variable specifies the path, file name, and file size for global temporary tablespace data files. File size is specified in KB, MB, or GB by appending K, M, or G to the size value. The sum of the sizes of the files must be slightly larger than 12MB.

To specify an alternate location for global temporary tablespace data files, configure the [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) variable at startup.

#### Session Temporary Tablespace Configuration

In MySQL 8.0.15 and earlier, session temporary tablespaces store user-created temporary tables and internal temporary tables created by the optimizer when **InnoDB** is configured as the on-disk storage engine for internal temporary tables ([**internal\_tmp\_disk\_storage\_engine=InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_internal_tmp_disk_storage_engine)). In MySQL 8.0.16 and later, the **InnoDB** storage engine is always used for internal temporary tables on disk.

The [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir) variable defines the location where **InnoDB** creates session temporary tablespaces. The default location is the #innodb\_temp directory in the data directory.

To specify an alternate location for session temporary tablespaces, configure the [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir) variable at startup. A fully qualified path or path relative to the data directory is permitted.

#### Page Size Configuration

The [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) option specifies the page size for all **InnoDB** tablespaces in a MySQL instance. This value is set when the instance is created and remains constant afterward. Valid values are 64KB, 32KB, 16KB (the default), 8KB, and 4KB. Alternatively, you can specify page size in bytes (65536, 32768, 16384, 8192, 4096).

The default page size of 16KB is appropriate for a wide range of workloads, particularly for queries involving table scans and DML operations involving bulk updates. Smaller page sizes might be more efficient for OLTP workloads involving many small writes, where contention can be an issue when a single page contains many rows. Smaller pages might also be efficient with SSD storage devices, which typically use small block sizes. Keeping the **InnoDB** page size close to the storage device block size minimizes the amount of unchanged data that is rewritten to disk.

#### Memory Configuration

MySQL allocates memory to various caches and buffers to improve performance of database operations. When allocating memory for **InnoDB**, always consider memory required by the operating system, memory allocated to other applications, and memory allocated for other MySQL buffers and caches. For example, if you use **MyISAM** tables, consider the amount of memory allocated for the key buffer ([**key\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_key_buffer_size)). For an overview of MySQL buffers and caches, see [Section 8.12.3.1, “How MySQL Uses Memory”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#memory-use).

Buffers specific to **InnoDB** are configured using the following parameters:

[**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) defines size of the buffer pool, which is the memory area that holds cached data for **InnoDB** tables, indexes, and other auxiliary buffers. The size of the buffer pool is important for system performance, and it is typically recommended that [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is configured to 50 to 75 percent of system memory. The default buffer pool size is 128MB. For additional guidance, see [Section 8.12.3.1, “How MySQL Uses Memory”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#memory-use). For information about how to configure **InnoDB** buffer pool size, see [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize). Buffer pool size can be configured at startup or dynamically.

On systems with a large amount of memory, you can improve concurrency by dividing the buffer pool into multiple buffer pool instances. The number of buffer pool instances is controlled by the by [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) option. By default, **InnoDB** creates one buffer pool instance. The number of buffer pool instances can be configured at startup. For more information, see [Section 15.8.3.2, “Configuring Multiple Buffer Pool Instances”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multiple-buffer-pools).

[**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) defines the size in bytes of the buffer that **InnoDB** uses to write to the log files on disk. The default size is 16MB. A large log buffer enables large transactions to run without a need to write the log to disk before the transactions commit. If you have transactions that update, insert, or delete many rows, you might consider increasing the size of the log buffer to save disk I/O. [**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) can be configured at startup. For related information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**Warning**

On 32-bit GNU/Linux x86, be careful not to set memory usage too high. **glibc** may permit the process heap to grow over thread stacks, which crashes your server. It is a risk if the memory allocated to the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process for global and per-thread buffers and caches is close to or exceeds 2GB.

A formula similar to the following that calculates global and per-thread memory allocation for MySQL can be used to estimate MySQL memory usage. You may need to modify the formula to account for buffers and caches in your MySQL version and configuration. For an overview of MySQL buffers and caches, see [Section 8.12.3.1, “How MySQL Uses Memory”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#memory-use).

innodb\_buffer\_pool\_size

+ key\_buffer\_size

+ max\_connections\*(sort\_buffer\_size+read\_buffer\_size+binlog\_cache\_size)

+ max\_connections\*2MB

Each thread uses a stack (often 2MB, but only 256KB in MySQL binaries provided by Oracle Corporation.) and in the worst case also uses **sort\_buffer\_size + read\_buffer\_size** additional memory.

On Linux, if the kernel is enabled for large page support, **InnoDB** can use large pages to allocate memory for its buffer pool. See [Section 8.12.3.2, “Enabling Large Page Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#large-page-support).

### 15.8.2 Configuring InnoDB for Read-Only Operation

You can query **InnoDB** tables where the MySQL data directory is on read-only media by enabling the [**--innodb-read-only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) configuration option at server startup.

#### How to Enable

To prepare an instance for read-only operation, make sure all the necessary information is [flushed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) to the data files before storing it on the read-only medium. Run the server with change buffering disabled ([**innodb\_change\_buffering=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering)) and do a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown).

To enable read-only mode for an entire MySQL instance, specify the following configuration options at server startup:

[**--innodb-read-only=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only)

If the instance is on read-only media such as a DVD or CD, or the /var directory is not writeable by all: [**--pid-file=*path\_on\_writeable\_media***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_pid_file) and [**--event-scheduler=disabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_event_scheduler)

[**--innodb-temp-data-file-path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path). This option specifies the path, file name, and file size for **InnoDB** temporary tablespace data files. The default setting is **ibtmp1:12M:autoextend**, which creates the ibtmp1 temporary tablespace data file in the data directory. To prepare an instance for read-only operation, set [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) to a location outside of the data directory. The path must be relative to the data directory. For example:

--innodb-temp-data-file-path=../../../tmp/ibtmp1:12M:autoextend

As of MySQL 8.0, enabling [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) prevents table creation and drop operations for all storage engines. These operations modify data dictionary tables in the **mysql** system database, but those tables use the **InnoDB** storage engine and cannot be modified when [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) is enabled. The same restriction applies to any operation that modifies data dictionary tables, such as [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) and [**ALTER TABLE *tbl\_name* ENGINE=*engine\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

In addition, other tables in the **mysql** system database use the **InnoDB** storage engine in MySQL 8.0. Making those tables read only results in restrictions on operations that modify them. For example, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user), [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke), and [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) operations are not permitted in read-only mode.

#### Usage Scenarios

This mode of operation is appropriate in situations such as:

Distributing a MySQL application, or a set of MySQL data, on a read-only storage medium such as a DVD or CD.

Multiple MySQL instances querying the same data directory simultaneously, typically in a data warehousing configuration. You might use this technique to avoid [bottlenecks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_bottleneck) that can occur with a heavily loaded MySQL instance, or you might use different configuration options for the various instances to tune each one for particular kinds of queries.

Querying data that has been put into a read-only state for security or data integrity reasons, such as archived backup data.

**Note**

This feature is mainly intended for flexibility in distribution and deployment, rather than raw performance based on the read-only aspect. See [Section 8.5.3, “Optimizing InnoDB Read-Only Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#innodb-performance-ro-txn) for ways to tune the performance of read-only queries, which do not require making the entire server read-only.

#### How It Works

When the server is run in read-only mode through the [**--innodb-read-only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) option, certain **InnoDB** features and components are reduced or turned off entirely:

No [change buffering](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffering) is done, in particular no merges from the change buffer. To make sure the change buffer is empty when you prepare the instance for read-only operation, disable change buffering ([**innodb\_change\_buffering=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering)) and do a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown) first.

There is no [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) phase at startup. The instance must have performed a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown) before being put into the read-only state.

Because the [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) is not used in read-only operation, you can set [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) to the smallest size possible (1 MB) before making the instance read-only.

Most background threads are turned off. I/O read threads remain, as well as I/O write threads and a page flush coordinator thread for writes to temporary files, which are permitted in read-only mode. A buffer pool resize thread also remains active to enable online resizing of the buffer pool.

Information about deadlocks, monitor output, and so on is not written to temporary files. As a consequence, [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) does not produce any output.

Changes to configuration option settings that would normally change the behavior of write operations, have no effect when the server is in read-only mode.

The [MVCC](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mvcc) processing to enforce [isolation levels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) is turned off. All queries read the latest version of a record, because update and deletes are not possible.

The [undo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_log) is not used. Disable any settings for the [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) and [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) configuration options.

### 15.8.3 InnoDB Buffer Pool Configuration

[15.8.3.1 Configuring InnoDB Buffer Pool Size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize)

[15.8.3.2 Configuring Multiple Buffer Pool Instances](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multiple-buffer-pools)

[15.8.3.3 Making the Buffer Pool Scan Resistant](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion)

[15.8.3.4 Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead)

[15.8.3.5 Configuring Buffer Pool Flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing)

[15.8.3.6 Saving and Restoring the Buffer Pool State](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool)

[15.8.3.7 Excluding Buffer Pool Pages from Core Files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-in-core-file)

This section provides configuration and tuning information for the **InnoDB** buffer pool.

#### 15.8.3.1 Configuring InnoDB Buffer Pool Size

You can configure **InnoDB** buffer pool size offline or while the server is running. Behavior described in this section applies to both methods. For additional information about configuring buffer pool size online, see [Configuring InnoDB Buffer Pool Size Online](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-online-resize).

When increasing or decreasing [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size), the operation is performed in chunks. Chunk size is defined by the [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) configuration option, which has a default of **128M**. For more information, see [Configuring InnoDB Buffer Pool Chunk Size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-chunk-size).

Buffer pool size must always be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). If you configure [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) to a value that is not equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances), buffer pool size is automatically adjusted to a value that is equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

In the following example, [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is set to **8G**, and [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) is set to **16**. [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) is **128M**, which is the default value.

**8G** is a valid [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) value because **8G** is a multiple of [**innodb\_buffer\_pool\_instances=16**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) \* [**innodb\_buffer\_pool\_chunk\_size=128M**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), which is **2G**.

shell> **mysqld --innodb-buffer-pool-size=8G --innodb-buffer-pool-instances=16**

mysql> **SELECT @@innodb\_buffer\_pool\_size/1024/1024/1024;**

+------------------------------------------+

| @@innodb\_buffer\_pool\_size/1024/1024/1024 |

+------------------------------------------+

| 8.000000000000 |

+------------------------------------------+

In this example, [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is set to **9G**, and [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) is set to **16**. [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) is **128M**, which is the default value. In this case, **9G** is not a multiple of [**innodb\_buffer\_pool\_instances=16**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) \* [**innodb\_buffer\_pool\_chunk\_size=128M**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), so [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is adjusted to **10G**, which is a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

shell> **mysqld --innodb-buffer-pool-size=9G --innodb-buffer-pool-instances=16**

mysql> **SELECT @@innodb\_buffer\_pool\_size/1024/1024/1024;**

+------------------------------------------+

| @@innodb\_buffer\_pool\_size/1024/1024/1024 |

+------------------------------------------+

| 10.000000000000 |

+------------------------------------------+

##### Configuring InnoDB Buffer Pool Chunk Size

[**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) can be increased or decreased in 1MB (1048576 byte) units but can only be modified at startup, in a command line string or in a MySQL configuration file.

Command line:

shell> **mysqld --innodb-buffer-pool-chunk-size=134217728**

Configuration file:

[mysqld]

innodb\_buffer\_pool\_chunk\_size=134217728

The following conditions apply when altering [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size):

If the new [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) value \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) is larger than the current buffer pool size when the buffer pool is initialized, [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) is truncated to [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) / [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

For example, if the buffer pool is initialized with a size of **2GB** (2147483648 bytes), **4** buffer pool instances, and a chunk size of **1GB** (1073741824 bytes), chunk size is truncated to a value equal to [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) / [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances), as shown below:

shell> **mysqld --innodb-buffer-pool-size=2147483648 --innodb-buffer-pool-instances=4**

**--innodb-buffer-pool-chunk-size=1073741824;**

mysql> **SELECT @@innodb\_buffer\_pool\_size;**

+---------------------------+

| @@innodb\_buffer\_pool\_size |

+---------------------------+

| 2147483648 |

+---------------------------+

mysql> **SELECT @@innodb\_buffer\_pool\_instances;**

+--------------------------------+

| @@innodb\_buffer\_pool\_instances |

+--------------------------------+

| 4 |

+--------------------------------+

# Chunk size was set to 1GB (1073741824 bytes) on startup but was

# truncated to innodb\_buffer\_pool\_size / innodb\_buffer\_pool\_instances

mysql> **SELECT @@innodb\_buffer\_pool\_chunk\_size;**

+---------------------------------+

| @@innodb\_buffer\_pool\_chunk\_size |

+---------------------------------+

| 536870912 |

+---------------------------------+

Buffer pool size must always be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). If you alter [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is automatically adjusted to a value that is equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). The adjustment occurs when the buffer pool is initialized. This behavior is demonstrated in the following example:

# The buffer pool has a default size of 128MB (134217728 bytes)

mysql> **SELECT @@innodb\_buffer\_pool\_size;**

+---------------------------+

| @@innodb\_buffer\_pool\_size |

+---------------------------+

| 134217728 |

+---------------------------+

# The chunk size is also 128MB (134217728 bytes)

mysql> **SELECT @@innodb\_buffer\_pool\_chunk\_size;**

+---------------------------------+

| @@innodb\_buffer\_pool\_chunk\_size |

+---------------------------------+

| 134217728 |

+---------------------------------+

# There is a single buffer pool instance

mysql> **SELECT @@innodb\_buffer\_pool\_instances;**

+--------------------------------+

| @@innodb\_buffer\_pool\_instances |

+--------------------------------+

| 1 |

+--------------------------------+

# Chunk size is decreased by 1MB (1048576 bytes) at startup

# (134217728 - 1048576 = 133169152):

shell> **mysqld --innodb-buffer-pool-chunk-size=133169152**

mysql> **SELECT @@innodb\_buffer\_pool\_chunk\_size;**

+---------------------------------+

| @@innodb\_buffer\_pool\_chunk\_size |

+---------------------------------+

| 133169152 |

+---------------------------------+

# Buffer pool size increases from 134217728 to 266338304

# Buffer pool size is automatically adjusted to a value that is equal to

# or a multiple of innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances

mysql> **SELECT @@innodb\_buffer\_pool\_size;**

+---------------------------+

| @@innodb\_buffer\_pool\_size |

+---------------------------+

| 266338304 |

+---------------------------+

This example demonstrates the same behavior but with multiple buffer pool instances:

# The buffer pool has a default size of 2GB (2147483648 bytes)

mysql> **SELECT @@innodb\_buffer\_pool\_size;**

+---------------------------+

| @@innodb\_buffer\_pool\_size |

+---------------------------+

| 2147483648 |

+---------------------------+

# The chunk size is .5 GB (536870912 bytes)

mysql> **SELECT @@innodb\_buffer\_pool\_chunk\_size;**

+---------------------------------+

| @@innodb\_buffer\_pool\_chunk\_size |

+---------------------------------+

| 536870912 |

+---------------------------------+

# There are 4 buffer pool instances

mysql> **SELECT @@innodb\_buffer\_pool\_instances;**

+--------------------------------+

| @@innodb\_buffer\_pool\_instances |

+--------------------------------+

| 4 |

+--------------------------------+

# Chunk size is decreased by 1MB (1048576 bytes) at startup

# (536870912 - 1048576 = 535822336):

shell> **mysqld --innodb-buffer-pool-chunk-size=535822336**

mysql> **SELECT @@innodb\_buffer\_pool\_chunk\_size;**

+---------------------------------+

| @@innodb\_buffer\_pool\_chunk\_size |

+---------------------------------+

| 535822336 |

+---------------------------------+

# Buffer pool size increases from 2147483648 to 4286578688

# Buffer pool size is automatically adjusted to a value that is equal to

# or a multiple of innodb\_buffer\_pool\_chunk\_size \* innodb\_buffer\_pool\_instances

mysql> **SELECT @@innodb\_buffer\_pool\_size;**

+---------------------------+

| @@innodb\_buffer\_pool\_size |

+---------------------------+

| 4286578688 |

+---------------------------+

Care should be taken when changing [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), as changing this value can increase the size of the buffer pool, as shown in the examples above. Before you change [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), calculate the effect on [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) to ensure that the resulting buffer pool size is acceptable.

**Note**

To avoid potential performance issues, the number of chunks ([**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) / [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size)) should not exceed 1000.

##### Configuring InnoDB Buffer Pool Size Online

The [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) configuration option can be set dynamically using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) statement, allowing you to resize the buffer pool without restarting the server. For example:

mysql> **SET GLOBAL innodb\_buffer\_pool\_size=402653184;**

**Note**

The buffer pool size must be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). Changing those variable settings requires restarting the server.

Active transactions and operations performed through **InnoDB** APIs should be completed before resizing the buffer pool. When initiating a resizing operation, the operation does not start until all active transactions are completed. Once the resizing operation is in progress, new transactions and operations that require access to the buffer pool must wait until the resizing operation finishes. The exception to the rule is that concurrent access to the buffer pool is permitted while the buffer pool is defragmented and pages are withdrawn when buffer pool size is decreased. A drawback of allowing concurrent access is that it could result in a temporary shortage of available pages while pages are being withdrawn.

**Note**

Nested transactions could fail if initiated after the buffer pool resizing operation begins.

##### Monitoring Online Buffer Pool Resizing Progress

The [**Innodb\_buffer\_pool\_resize\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_resize_status) reports buffer pool resizing progress. For example:

mysql> **SHOW STATUS WHERE Variable\_name='InnoDB\_buffer\_pool\_resize\_status';**

+----------------------------------+----------------------------------+

| Variable\_name | Value |

+----------------------------------+----------------------------------+

| Innodb\_buffer\_pool\_resize\_status | Resizing also other hash tables. |

+----------------------------------+----------------------------------+

Buffer pool resizing progress is also logged in the server error log. This example shows notes that are logged when increasing the size of the buffer pool:

[Note] InnoDB: Resizing buffer pool from 134217728 to 4294967296. (unit=134217728)

[Note] InnoDB: disabled adaptive hash index.

[Note] InnoDB: buffer pool 0 : 31 chunks (253952 blocks) was added.

[Note] InnoDB: buffer pool 0 : hash tables were resized.

[Note] InnoDB: Resized hash tables at lock\_sys, adaptive hash index, dictionary.

[Note] InnoDB: completed to resize buffer pool from 134217728 to 4294967296.

[Note] InnoDB: re-enabled adaptive hash index.

This example shows notes that are logged when decreasing the size of the buffer pool:

[Note] InnoDB: Resizing buffer pool from 4294967296 to 134217728. (unit=134217728)

[Note] InnoDB: disabled adaptive hash index.

[Note] InnoDB: buffer pool 0 : start to withdraw the last 253952 blocks.

[Note] InnoDB: buffer pool 0 : withdrew 253952 blocks from free list. tried to relocate 0 pages.

(253952/253952)

[Note] InnoDB: buffer pool 0 : withdrawn target 253952 blocks.

[Note] InnoDB: buffer pool 0 : 31 chunks (253952 blocks) was freed.

[Note] InnoDB: buffer pool 0 : hash tables were resized.

[Note] InnoDB: Resized hash tables at lock\_sys, adaptive hash index, dictionary.

[Note] InnoDB: completed to resize buffer pool from 4294967296 to 134217728.

[Note] InnoDB: re-enabled adaptive hash index.

##### Online Buffer Pool Resizing Internals

The resizing operation is performed by a background thread. When increasing the size of the buffer pool, the resizing operation:

Adds pages in **chunks** (chunk size is defined by [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size))

Coverts hash tables, lists, and pointers to use new addresses in memory

Adds new pages to the free list

While these operations are in progress, other threads are blocked from accessing the buffer pool.

When decreasing the size of the buffer pool, the resizing operation:

Defragments the buffer pool and withdraws (frees) pages

Removes pages in **chunks** (chunk size is defined by [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size))

Converts hash tables, lists, and pointers to use new addresses in memory

Of these operations, only defragmenting the buffer pool and withdrawing pages allow other threads to access to the buffer pool concurrently.

#### 15.8.3.2 Configuring Multiple Buffer Pool Instances

For systems with buffer pools in the multi-gigabyte range, dividing the buffer pool into separate instances can improve concurrency, by reducing contention as different threads read and write to cached pages. This feature is typically intended for systems with a [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) size in the multi-gigabyte range. Multiple buffer pool instances are configured using the [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) configuration option, and you might also adjust the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) value.

When the **InnoDB** buffer pool is large, many data requests can be satisfied by retrieving from memory. You might encounter bottlenecks from multiple threads trying to access the buffer pool at once. You can enable multiple buffer pools to minimize this contention. Each page that is stored in or read from the buffer pool is assigned to one of the buffer pools randomly, using a hashing function. Each buffer pool manages its own free lists, flush lists, LRUs, and all other data structures connected to a buffer pool. Prior to MySQL 8.0, each buffer pool was protected by its own buffer pool mutex. In MySQL 8.0 and later, the buffer pool mutex was replaced by several list and hash protecting mutexes, to reduce contention.

To enable multiple buffer pool instances, set the **innodb\_buffer\_pool\_instances** configuration option to a value greater than 1 (the default) up to 64 (the maximum). This option takes effect only when you set **innodb\_buffer\_pool\_size** to a size of 1GB or more. The total size you specify is divided among all the buffer pools. For best efficiency, specify a combination of [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) and [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) so that each buffer pool instance is at least 1GB.

For information about modifying **InnoDB** buffer pool size, see [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize).

#### 15.8.3.3 Making the Buffer Pool Scan Resistant

Rather than using a strict [LRU](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lru) algorithm, **InnoDB** uses a technique to minimize the amount of data that is brought into the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) and never accessed again. The goal is to make sure that frequently accessed (“hot”) pages remain in the buffer pool, even as [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) and [full table scans](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_full_table_scan) bring in new blocks that might or might not be accessed afterward.

Newly read blocks are inserted into the middle of the LRU list. All newly read pages are inserted at a location that by default is **3/8** from the tail of the LRU list. The pages are moved to the front of the list (the most-recently used end) when they are accessed in the buffer pool for the first time. Thus, pages that are never accessed never make it to the front portion of the LRU list, and “age out” sooner than with a strict LRU approach. This arrangement divides the LRU list into two segments, where the pages downstream of the insertion point are considered “old” and are desirable victims for LRU eviction.

For an explanation of the inner workings of the **InnoDB** buffer pool and specifics about the LRU algorithm, see [Section 15.5.1, “Buffer Pool”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool).

You can control the insertion point in the LRU list and choose whether **InnoDB** applies the same optimization to blocks brought into the buffer pool by table or index scans. The configuration parameter [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) controls the percentage of “old” blocks in the LRU list. The default value of [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) is **37**, corresponding to the original fixed ratio of 3/8. The value range is **5** (new pages in the buffer pool age out very quickly) to **95** (only 5% of the buffer pool is reserved for hot pages, making the algorithm close to the familiar LRU strategy).

The optimization that keeps the buffer pool from being churned by read-ahead can avoid similar problems due to table or index scans. In these scans, a data page is typically accessed a few times in quick succession and is never touched again. The configuration parameter [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) specifies the time window (in milliseconds) after the first access to a page during which it can be accessed without being moved to the front (most-recently used end) of the LRU list. The default value of [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) is **1000**. Increasing this value makes more and more blocks likely to age out faster from the buffer pool.

Both [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) and [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) can be specified in the MySQL option file (**my.cnf** or **my.ini**) or changed at runtime with the [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. Changing the value at runtime requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

To help you gauge the effect of setting these parameters, the **SHOW ENGINE INNODB STATUS** command reports buffer pool statistics. For details, see [Monitoring the Buffer Pool Using the InnoDB Standard Monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-monitoring).

Because the effects of these parameters can vary widely based on your hardware configuration, your data, and the details of your workload, always benchmark to verify the effectiveness before changing these settings in any performance-critical or production environment.

In mixed workloads where most of the activity is OLTP type with periodic batch reporting queries which result in large scans, setting the value of [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) during the batch runs can help keep the working set of the normal workload in the buffer pool.

When scanning large tables that cannot fit entirely in the buffer pool, setting [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) to a small value keeps the data that is only read once from consuming a significant portion of the buffer pool. For example, setting **innodb\_old\_blocks\_pct=5** restricts this data that is only read once to 5% of the buffer pool.

When scanning small tables that do fit into memory, there is less overhead for moving pages around within the buffer pool, so you can leave [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) at its default value, or even higher, such as **innodb\_old\_blocks\_pct=50**.

The effect of the [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) parameter is harder to predict than the [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) parameter, is relatively small, and varies more with the workload. To arrive at an optimal value, conduct your own benchmarks if the performance improvement from adjusting [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) is not sufficient.

#### 15.8.3.4 Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)

A [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) request is an I/O request to prefetch multiple pages in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) asynchronously, in anticipation of impending need for these pages. The requests bring in all the pages in one [extent](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_extent). **InnoDB** uses two read-ahead algorithms to improve I/O performance:

**Linear** read-ahead is a technique that predicts what pages might be needed soon based on pages in the buffer pool being accessed sequentially. You control when **InnoDB** performs a read-ahead operation by adjusting the number of sequential page accesses required to trigger an asynchronous read request, using the configuration parameter [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold). Before this parameter was added, **InnoDB** would only calculate whether to issue an asynchronous prefetch request for the entire next extent when it read the last page of the current extent.

The configuration parameter [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) controls how sensitive **InnoDB** is in detecting patterns of sequential page access. If the number of pages read sequentially from an extent is greater than or equal to [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold), **InnoDB** initiates an asynchronous read-ahead operation of the entire following extent. [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) can be set to any value from 0-64. The default value is 56. The higher the value, the more strict the access pattern check. For example, if you set the value to 48, **InnoDB** triggers a linear read-ahead request only when 48 pages in the current extent have been accessed sequentially. If the value is 8, **InnoDB** triggers an asynchronous read-ahead even if as few as 8 pages in the extent are accessed sequentially. You can set the value of this parameter in the MySQL [configuration file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_configuration_file), or change it dynamically with the [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, which requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

**Random** read-ahead is a technique that predicts when pages might be needed soon based on pages already in the buffer pool, regardless of the order in which those pages were read. If 13 consecutive pages from the same extent are found in the buffer pool, **InnoDB** asynchronously issues a request to prefetch the remaining pages of the extent. To enable this feature, set the configuration variable [**innodb\_random\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_random_read_ahead) to **ON**.

The **SHOW ENGINE INNODB STATUS** command displays statistics to help you evaluate the effectiveness of the read-ahead algorithm. Statistics include counter information for the following global status variables:

[**Innodb\_buffer\_pool\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead)

[**Innodb\_buffer\_pool\_read\_ahead\_evicted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead_evicted)

[**Innodb\_buffer\_pool\_read\_ahead\_rnd**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead_rnd)

This information can be useful when fine-tuning the [**innodb\_random\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_random_read_ahead) setting.

For more information about I/O performance, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio) and [Section 8.12.1, “Optimizing Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#disk-issues).

#### 15.8.3.5 Configuring Buffer Pool Flushing

**InnoDB** performs certain tasks in the background, including flushing of dirty pages from the buffer pool. Dirty pages are those that have been modified but are not yet written to the data files on disk.

In MySQL 8.0, buffer pool flushing is performed by page cleaner threads. The number of page cleaner threads is controlled by the [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) variable, which has a default value of 4. However, if the number of page cleaner threads exceeds the number of buffer pool instances, [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) is automatically set to the same value as [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

Buffer pool flushing is initiated when the percentage of dirty pages reaches the low water mark value defined by the [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm) variable. The default low water mark is 10% of buffer pool pages. A [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm) value of 0 disables this early flushing behaviour.

The purpose of the [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm) threshold is to control the percentage dirty pages in the buffer pool and to prevent the amount of dirty pages from reaching the threshold defined by the [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) variable, which has a default value of 90. **InnoDB** aggressively flushes buffer pool pages if the percentage of dirty pages in the buffer pool reaches the [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) threshold.

When configuring [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm), the value should always be lower than the [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) value.

Additional variables permit fine-tuning of buffer pool flushing behavior:

The [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors) variable defines whether flushing a page from the buffer pool also flushes other dirty pages in the same extent.

The default setting of 0 disables [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors). Dirty pages in the same extent are not flushed. This setting is recommended for non-rotational storage (SSD) devices where seek time is not a significant factor.

A setting of 1 flushes contiguous dirty pages in the same extent.

A setting of 2 flushes dirty pages in the same extent.

When table data is stored on a traditional [HDD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_hdd) storage device, flushing neighbor pages in one operation reduces I/O overhead (primarily for disk seek operations) compared to flushing individual pages at different times. For table data stored on [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd), seek time is not a significant factor and you can disable this setting to spread out write operations.

The [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) variable specifies, per buffer pool instance, how far down the buffer pool LRU list the page cleaner thread scans looking for dirty pages to flush. This is a background operation performed by a page cleaner thread once per second.

A setting smaller than the default is generally suitable for most workloads. A value that is significantly higher than necessary may impact performance. Only consider increasing the value if you have spare I/O capacity under a typical workload. Conversely, if a write-intensive workload saturates your I/O capacity, decrease the value, especially in the case of a large buffer pool.

When tuning [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth), start with a low value and configure the setting upward with the goal of rarely seeing zero free pages. Also, consider adjusting [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) when changing the number of buffer pool instances, since [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) defines the amount of work performed by the page cleaner thread each second.

The [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors) and [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) variables are primarily intended for write-intensive workloads. With heavy DML activity, flushing can fall behind if it is not aggressive enough, or disk writes can saturate I/O capacity if flushing is too aggressive. The ideal settings depend on your workload, data access patterns, and storage configuration (for example, whether data is stored on HDD or SSD devices).

##### Adaptive Flushing

**InnoDB** uses an adaptive flushing algorithm to dynamically adjust the rate of flushing based on the speed of redo log generation and the current rate of flushing. The intent is to smooth overall performance by ensuring that flushing activity keeps pace with the current workload. Automatically adjusting the flushing rate helps avoid sudden dips in throughput that can occur when bursts of I/O activity due to buffer pool flushing affects the I/O capacity available for ordinary read and write activity.

Sharp checkpoints, which are typically associated with write-intensive workloads that generate a lot of redo entries, can cause a sudden change in throughput, for example. A sharp checkpoint occurs when **InnoDB** wants to reuse a portion of a log file. Before doing so, all dirty pages with redo entries in that portion of the log file must be flushed. If log files become full, a sharp checkpoint occurs, causing a temporary reduction in throughput. This scenario can occur even if [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) threshold is not reached.

The adaptive flushing algorithm helps avoid such scenarios by tracking the number of dirty pages in the buffer pool and the rate at which redo log records are being generated. Based on this information, it decides how many dirty pages to flush from the buffer pool each second, which permits it to manage sudden changes in workload.

The [**innodb\_adaptive\_flushing\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing_lwm) variable defines a low water mark for redo log capacity. When that threshold is crossed, adaptive flushing is enabled, even if the [**innodb\_adaptive\_flushing**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing) variable is disabled.

Internal benchmarking has shown that the algorithm not only maintains throughput over time, but can also improve overall throughput significantly. However, adaptive flushing can affect the I/O pattern of a workload significantly and may not be appropriate in all cases. It gives the most benefit when the redo log is in danger of filling up. If adaptive flushing is not appropriate to the characteristics of your workload, you can disable it. Adaptive flushing controlled by the [**innodb\_adaptive\_flushing**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing) variable, which is enabled by default.

[**innodb\_flushing\_avg\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flushing_avg_loops) defines the number of iterations that **InnoDB** keeps the previously calculated snapshot of the flushing state, controlling how quickly adaptive flushing responds to foreground workload changes. A high [**innodb\_flushing\_avg\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flushing_avg_loops) value means that **InnoDB** keeps the previously calculated snapshot longer, so adaptive flushing responds more slowly. When setting a high value it is important to ensure that redo log utilization does not reach 75% (the hardcoded limit at which asynchronous flushing starts), and that the [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) threshold keeps the number of dirty pages to a level that is appropriate for the workload.

Systems with consistent workloads, a large log file size ([**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size)), and small spikes that do not reach 75% log space utilization should use a high [**innodb\_flushing\_avg\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flushing_avg_loops) value to keep flushing as smooth as possible. For systems with extreme load spikes or log files that do not provide a lot of space, a smaller value allows flushing to closely track workload changes, and helps to avoid reaching 75% log space utilization.

Be aware that if flushing falls behind, the rate of buffer pool flushing can exceed the I/O capacity available to **InnoDB**, as defined by [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting. The [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) value defines an upper limit on I/O capacity in such situations, so that a spike in I/O activity does not consume the entire I/O capacity of the server.

The [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting is applicable to all buffer pool instances. When dirty pages are flushed, I/O capacity is divided equally among buffer pool instances.

##### Limiting Buffer Flushing During Idle Periods

As of MySQL 8.0.18, you can use the [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) variable to limit the rate of buffer pool flushing during idle periods, which are periods of time that database pages are not modified. The [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) value is a percentage of the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting, which defines the number of I/O operations per second available to **InnoDB**. The default [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) value is 100, which is 100 percent of the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting. To limit flushing during idle periods, define an [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) value less than 100.

Limiting page flushing during idle periods can help extend the life of solid state storage devices. Side effects of limiting page flushing during idle periods may include a longer shutdown time following a lengthy idle period, and a longer recovery period should a server failure occur.

#### 15.8.3.6 Saving and Restoring the Buffer Pool State

To reduce the [warmup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_warm_up) period after restarting the server, **InnoDB** saves a percentage of the most recently used pages for each buffer pool at server shutdown and restores these pages at server startup. The percentage of recently used pages that is stored is defined by the [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) configuration option.

After restarting a busy server, there is typically a warmup period with steadily increasing throughput, as disk pages that were in the buffer pool are brought back into memory (as the same data is queried, updated, and so on). The ability to restore the buffer pool at startup shortens the warmup period by reloading disk pages that were in the buffer pool before the restart rather than waiting for DML operations to access corresponding rows. Also, I/O requests can be performed in large batches, making the overall I/O faster. Page loading happens in the background, and does not delay database startup.

In addition to saving the buffer pool state at shutdown and restoring it at startup, you can save and restore the buffer pool state at any time, while the server is running. For example, you can save the state of the buffer pool after reaching a stable throughput under a steady workload. You could also restore the previous buffer pool state after running reports or maintenance jobs that bring data pages into the buffer pool that are only requited for those operations, or after running some other non-typical workload.

Even though a buffer pool can be many gigabytes in size, the buffer pool data that **InnoDB** saves to disk is tiny by comparison. Only tablespace IDs and page IDs necessary to locate the appropriate pages are saved to disk. This information is derived from the [**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table) **INFORMATION\_SCHEMA** table. By default, tablespace ID and page ID data is saved in a file named ib\_buffer\_pool, which is saved to the **InnoDB** data directory. The file name and location can be modified using the [**innodb\_buffer\_pool\_filename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_filename) configuration parameter.

Because data is cached in and aged out of the buffer pool as it is with regular database operations, there is no problem if the disk pages are recently updated, or if a DML operation involves data that has not yet been loaded. The loading mechanism skips requested pages that no longer exist.

The underlying mechanism involves a background thread that is dispatched to perform the dump and load operations.

Disk pages from compressed tables are loaded into the buffer pool in their compressed form. Pages are uncompressed as usual when page contents are accessed during DML operations. Because uncompressing pages is a CPU-intensive process, it is more efficient for concurrency to perform the operation in a connection thread rather than in the single thread that performs the buffer pool restore operation.

Operations related to saving and restoring the buffer pool state are described in the following topics:

[Configuring the Dump Percentage for Buffer Pool Pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-dump-pct)

[Saving the Buffer Pool State at Shutdown and Restoring it at Startup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-offline)

[Saving and Restoring the Buffer Pool State Online](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-online)

[Displaying Buffer Pool Dump Progress](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-dump-progress)

[Displaying Buffer Pool Load Progress](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-load-progress)

[Aborting a Buffer Pool Load Operation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool-abort-load)

[Monitoring Buffer Pool Load Progress Using Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-buffer-pool-load-performance-schema)

##### Configuring the Dump Percentage for Buffer Pool Pages

Before dumping pages from the buffer pool, you can configure the percentage of most-recently-used buffer pool pages that you want to dump by setting the [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) option. If you plan to dump buffer pool pages while the server is running, you can configure the option dynamically:

SET GLOBAL innodb\_buffer\_pool\_dump\_pct=40;

If you plan to dump buffer pool pages at server shutdown, set [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) in your configuration file.

[mysqld]

innodb\_buffer\_pool\_dump\_pct=40

The [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) default value is 25 (dump 25% of most-recently-used pages).

##### Saving the Buffer Pool State at Shutdown and Restoring it at Startup

To save the state of the buffer pool at server shutdown, issue the following statement prior to shutting down the server:

SET GLOBAL innodb\_buffer\_pool\_dump\_at\_shutdown=ON;

[**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) is enabled by default.

To restore the buffer pool state at server startup, specify the **--innodb-buffer-pool-load-at-startup** option when starting the server:

mysqld --innodb-buffer-pool-load-at-startup=ON;

[**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) is enabled by default.

##### Saving and Restoring the Buffer Pool State Online

To save the state of the buffer pool while MySQL server is running, issue the following statement:

SET GLOBAL innodb\_buffer\_pool\_dump\_now=ON;

To restore the buffer pool state while MySQL is running, issue the following statement:

SET GLOBAL innodb\_buffer\_pool\_load\_now=ON;

##### Displaying Buffer Pool Dump Progress

To display progress when saving the buffer pool state to disk, issue the following statement:

SHOW STATUS LIKE 'Innodb\_buffer\_pool\_dump\_status';

If the operation has not yet started, “not started” is returned. If the operation is complete, the completion time is printed (e.g. Finished at 110505 12:18:02). If the operation is in progress, status information is provided (e.g. Dumping buffer pool 5/7, page 237/2873).

##### Displaying Buffer Pool Load Progress

To display progress when loading the buffer pool, issue the following statement:

SHOW STATUS LIKE 'Innodb\_buffer\_pool\_load\_status';

If the operation has not yet started, “not started” is returned. If the operation is complete, the completion time is printed (e.g. Finished at 110505 12:23:24). If the operation is in progress, status information is provided (e.g. Loaded 123/22301 pages).

##### Aborting a Buffer Pool Load Operation

To abort a buffer pool load operation, issue the following statement:

SET GLOBAL innodb\_buffer\_pool\_load\_abort=ON;

##### Monitoring Buffer Pool Load Progress Using Performance Schema

You can monitor buffer pool load progress using [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html).

The following example demonstrates how to enable the **stage/innodb/buffer pool load** stage event instrument and related consumer tables to monitor buffer pool load progress.

For information about buffer pool dump and load procedures used in this example, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool). For information about Performance Schema stage event instruments and related consumers, see [Section 27.12.5, “Performance Schema Stage Event Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-tables).

Enable the **stage/innodb/buffer pool load** instrument:

mysql> **UPDATE performance\_schema.setup\_instruments SET ENABLED = 'YES'**

**WHERE NAME LIKE 'stage/innodb/buffer%';**

Enable the stage event consumer tables, which include [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table), [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table), and [**events\_stages\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-long-table).

mysql> **UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES'**

**WHERE NAME LIKE '%stages%';**

Dump the current buffer pool state by enabling [**innodb\_buffer\_pool\_dump\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_now).

mysql> **SET GLOBAL innodb\_buffer\_pool\_dump\_now=ON;**

Check the buffer pool dump status to ensure that the operation has completed.

mysql> **SHOW STATUS LIKE 'Innodb\_buffer\_pool\_dump\_status'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Variable\_name: Innodb\_buffer\_pool\_dump\_status

Value: Buffer pool(s) dump completed at 150202 16:38:58

Load the buffer pool by enabling [**innodb\_buffer\_pool\_load\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_now):

mysql> **SET GLOBAL innodb\_buffer\_pool\_load\_now=ON;**

Check the current status of the buffer pool load operation by querying the Performance Schema [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table. The **WORK\_COMPLETED** column shows the number of buffer pool pages loaded. The **WORK\_ESTIMATED** column provides an estimate of the remaining work, in pages.

mysql> **SELECT EVENT\_NAME, WORK\_COMPLETED, WORK\_ESTIMATED**

**FROM performance\_schema.events\_stages\_current;**

+-------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+-------------------------------+----------------+----------------+

| stage/innodb/buffer pool load | 5353 | 7167 |

+-------------------------------+----------------+----------------+

The [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table returns an empty set if the buffer pool load operation has completed. In this case, you can check the [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table) table to view data for the completed event. For example:

mysql> **SELECT EVENT\_NAME, WORK\_COMPLETED, WORK\_ESTIMATED**

**FROM performance\_schema.events\_stages\_history;**

+-------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+-------------------------------+----------------+----------------+

| stage/innodb/buffer pool load | 7167 | 7167 |

+-------------------------------+----------------+----------------+

**Note**

You can also monitor buffer pool load progress using Performance Schema when loading the buffer pool at startup using [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup). In this case, the **stage/innodb/buffer pool load** instrument and related consumers must be enabled at startup. For more information, see [Section 27.3, “Performance Schema Startup Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-startup-configuration).

#### 15.8.3.7 Excluding Buffer Pool Pages from Core Files

A core file records the status and memory image of a running process. Because the buffer pool resides in main memory, and the memory image of a running process is dumped to the core file, systems with large buffer pools can produce large core files when the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process dies.

Large core files can be problematic for a number of reasons including the time it takes to write them, the amount of disk space they consume, and the challenges associated with transferring large files.

To reduce core file size, you can disable the [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable to omit buffer pool pages from core dumps. The [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable was introduced in MySQL 8.0.14 and is enabled by default.

Excluding buffer pool pages may also be desirable from a security perspective if you have concerns about dumping database pages to core files that may be shared inside or outside of your organization for debugging purposes.

**Note**

Access to the data present in buffer pool pages at the time the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process died may be beneficial in some debugging scenarios. If in doubt whether to include or exclude buffer pool pages, consult MySQL Support.

Disabling [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) takes effect only if the [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) variable is enabled and the operating system supports the **MADV\_DONTDUMP** non-POSIX extension to the [madvise()](http://man7.org/linux/man-pages/man2/madvise.2.html) system call, which is supported in Linux 3.4 and later. The **MADV\_DONTDUMP** extension causes pages in a specified range to be excluded from core dumps.

Assuming the operating system supports the **MADV\_DONTDUMP** extension, start the server with the [--core-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_core-file) and [**--innodb-buffer-pool-in-core-file=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) options to generate core files without buffer pool pages.

shell> mysqld --core-file --innodb-buffer-pool-in-core-file=OFF

The [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) variable is read only and disabled by default. It is enabled by specifying the [--core-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_core-file) option at startup. The [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable is dynamic. It can be specified at startup or configured at runtime using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement.

mysql> SET GLOBAL innodb\_buffer\_pool\_in\_core\_file=OFF;

If the [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable is disabled but **MADV\_DONTDUMP** is not supported by the operating system, or an **madvise()** failure occurs, a warning is written to the MySQL server error log and the [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) variable is disabled to prevent writing core files that unintentionally include buffer pool pages. If the read-only [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) variable becomes disabled, the server must be restarted to enable it again.

The following table shows configuration and **MADV\_DONTDUMP** support scenarios that determine whether core files are generated and whether they include buffer pool pages.

**Table 15.4 Core File Configuration Scenarios**

| [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file)**variable** | [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file)**variable** | **madvise() MADV\_DONTDUMP Support** | **Outcome** |
| --- | --- | --- | --- |
| **OFF (default)** | Not relevant to outcome | Not relevant to outcome | Core file is not generated |
| **ON** | ON (default) | Not relevant to outcome | Core file is generated with buffer pool pages |
| **ON** | OFF | Yes | Core file is generated without buffer pool pages |
| **ON** | OFF | No | Core file is not generated, [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) is disabled, and a warning is written to the server error log |

The reduction in core file size achieved by disabling the [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable depends on the size of the buffer pool, but it is also affected by the **InnoDB** page size. A smaller page size means more pages are required for the same amount of data, and more pages means more page metadata. The following table provides size reduction examples that you might see for a 1GB buffer pool with different pages sizes.

**Table 15.5 Core File Size with Buffer Pool Pages Included and Excluded**

| [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size)**Setting** | **Buffer Pool Pages Included (**[**innodb\_buffer\_pool\_in\_core\_file=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file)**)** | **Buffer Pool Pages Excluded (**[**innodb\_buffer\_pool\_in\_core\_file=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file)**)** |
| --- | --- | --- |
| **4KB** | 2.1GB | 0.9GB |
| **64KB** | 1.7GB | 0.7GB |

### 15.8.4 Configuring Thread Concurrency for InnoDB

**InnoDB** uses operating system [threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_thread) to process requests from user transactions. (Transactions may issue many requests to **InnoDB** before they commit or roll back.) On modern operating systems and servers with multi-core processors, where context switching is efficient, most workloads run well without any limit on the number of concurrent threads.

In situations where it is helpful to minimize context switching between threads, **InnoDB** can use a number of techniques to limit the number of concurrently executing operating system threads (and thus the number of requests that are processed at any one time). When **InnoDB** receives a new request from a user session, if the number of threads concurrently executing is at a pre-defined limit, the new request sleeps for a short time before it tries again. A request that cannot be rescheduled after the sleep is put in a first-in/first-out queue and eventually is processed. Threads waiting for locks are not counted in the number of concurrently executing threads.

You can limit the number of concurrent threads by setting the configuration parameter [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency). Once the number of executing threads reaches this limit, additional threads sleep for a number of microseconds, set by the configuration parameter [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay), before being placed into the queue.

You can set the configuration option [**innodb\_adaptive\_max\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_max_sleep_delay) to the highest value you would allow for [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay), and **InnoDB** automatically adjusts [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) up or down depending on the current thread-scheduling activity. This dynamic adjustment helps the thread scheduling mechanism to work smoothly during times when the system is lightly loaded and when it is operating near full capacity.

The default value for [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) and the implied default limit on the number of concurrent threads has been changed in various releases of MySQL and **InnoDB**. The default value of [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is **0**, so that by default there is no limit on the number of concurrently executing threads.

**InnoDB** causes threads to sleep only when the number of concurrent threads is limited. When there is no limit on the number of threads, all contend equally to be scheduled. That is, if [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is **0**, the value of [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) is ignored.

When there is a limit on the number of threads (when [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is > 0), **InnoDB** reduces context switching overhead by permitting multiple requests made during the execution of a single SQL statement to enter **InnoDB** without observing the limit set by [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency). Since an SQL statement (such as a join) may comprise multiple row operations within **InnoDB**, **InnoDB** assigns a specified number of “tickets” that allow a thread to be scheduled repeatedly with minimal overhead.

When a new SQL statement starts, a thread has no tickets, and it must observe [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency). Once the thread is entitled to enter **InnoDB**, it is assigned a number of tickets that it can use for subsequently entering **InnoDB** to perform row operations. If the tickets run out, the thread is evicted, and [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is observed again which may place the thread back into the first-in/first-out queue of waiting threads. When the thread is once again entitled to enter **InnoDB**, tickets are assigned again. The number of tickets assigned is specified by the global option [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets), which is 5000 by default. A thread that is waiting for a lock is given one ticket once the lock becomes available.

The correct values of these variables depend on your environment and workload. Try a range of different values to determine what value works for your applications. Before limiting the number of concurrently executing threads, review configuration options that may improve the performance of **InnoDB** on multi-core and multi-processor computers, such as [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index).

For general performance information about MySQL thread handling, see [Section 5.1.12.1, “Connection Interfaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#connection-interfaces).

### 15.8.5 Configuring the Number of Background InnoDB I/O Threads

**InnoDB** uses background [threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_thread) to service various types of I/O requests. You can configure the number of background threads that service read and write I/O on data pages using the [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) and [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) configuration parameters. These parameters signify the number of background threads used for read and write requests, respectively. They are effective on all supported platforms. You can set values for these parameters in the MySQL option file (**my.cnf** or **my.ini**); you cannot change values dynamically. The default value for these parameters is **4** and permissible values range from **1-64**.

The purpose of these configuration options to make **InnoDB** more scalable on high end systems. Each background thread can handle up to 256 pending I/O requests. A major source of background I/O is [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) requests. **InnoDB** tries to balance the load of incoming requests in such way that most background threads share work equally. **InnoDB** also attempts to allocate read requests from the same extent to the same thread, to increase the chances of coalescing the requests. If you have a high end I/O subsystem and you see more than 64 × [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) pending read requests in **SHOW ENGINE INNODB STATUS** output, you might improve performance by increasing the value of [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads).

On Linux systems, **InnoDB** uses the asynchronous I/O subsystem by default to perform read-ahead and write requests for data file pages, which changes the way that **InnoDB** background threads service these types of I/O requests. For more information, see [Section 15.8.6, “Using Asynchronous I/O on Linux”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-linux-native-aio).

For more information about **InnoDB** I/O performance, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

### 15.8.6 Using Asynchronous I/O on Linux

**InnoDB** uses the asynchronous I/O subsystem (native AIO) on Linux to perform read-ahead and write requests for data file pages. This behavior is controlled by the [**innodb\_use\_native\_aio**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_native_aio) configuration option, which applies to Linux systems only and is enabled by default. On other Unix-like systems, **InnoDB** uses synchronous I/O only. Historically, **InnoDB** only used asynchronous I/O on Windows systems. Using the asynchronous I/O subsystem on Linux requires the **libaio** library.

With synchronous I/O, query threads queue I/O requests, and **InnoDB** background threads retrieve the queued requests one at a time, issuing a synchronous I/O call for each. When an I/O request is completed and the I/O call returns, the **InnoDB** background thread that is handling the request calls an I/O completion routine and returns to process the next request. The number of requests that can be processed in parallel is ***n***, where ***n*** is the number of **InnoDB** background threads. The number of **InnoDB** background threads is controlled by [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) and [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads). See [Section 15.8.5, “Configuring the Number of Background InnoDB I/O Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-multiple_io_threads).

With native AIO, query threads dispatch I/O requests directly to the operating system, thereby removing the limit imposed by the number of background threads. **InnoDB** background threads wait for I/O events to signal completed requests. When a request is completed, a background thread calls an I/O completion routine and resumes waiting for I/O events.

The advantage of native AIO is scalability for heavily I/O-bound systems that typically show many pending reads/writes in **SHOW ENGINE INNODB STATUS\G** output. The increase in parallel processing when using native AIO means that the type of I/O scheduler or properties of the disk array controller have a greater influence on I/O performance.

A potential disadvantage of native AIO for heavily I/O-bound systems is lack of control over the number of I/O write requests dispatched to the operating system at once. Too many I/O write requests dispatched to the operating system for parallel processing could, in some cases, result in I/O read starvation, depending on the amount of I/O activity and system capabilities.

If a problem with the asynchronous I/O subsystem in the OS prevents **InnoDB** from starting, you can start the server with [**innodb\_use\_native\_aio=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_native_aio). This option may also be disabled automatically during startup if **InnoDB** detects a potential problem such as a combination of **tmpdir** location, **tmpfs** file system, and Linux kernel that does not support asynchronous I/O on **tmpfs**.

### 15.8.7 Configuring InnoDB I/O Capacity

The **InnoDB** master thread and other threads perform various tasks in the background, most of which are I/O related, such as flushing dirty pages from the buffer pool and writing changes from the change buffer to the appropriate secondary indexes. **InnoDB** attempts to perform these tasks in a way that does not adversely affect the normal working of the server. It tries to estimate the available I/O bandwidth and tune its activities to take advantage of available capacity.

The [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) variable defines the overall I/O capacity available to **InnoDB**. It should be set to approximately the number of I/O operations that the system can perform per second (IOPS). When [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) is set, **InnoDB** estimates the I/O bandwidth available for background tasks based on the set value.

You can set [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) to a value of 100 or greater. The default value is **200**. Typically, values around 100 are appropriate for consumer-level storage devices, such as hard drives up to 7200 RPMs. Faster hard drives, RAID configurations, and solid state drives (SSDs) benefit from higher values.

Ideally, keep the setting as low as practical, but not so low that background activities fall behind. If the value is too high, data is removed from the buffer pool and change buffer too quickly for caching to provide a significant benefit. For busy systems capable of higher I/O rates, you can set a higher value to help the server handle the background maintenance work associated with a high rate of row changes. Generally, you can increase the value as a function of the number of drives used for **InnoDB** I/O. For example, you can increase the value on systems that use multiple disks or SSDs.

The default setting of 200 is generally sufficient for a lower-end SSD. For a higher-end, bus-attached SSD, consider a higher setting such as 1000, for example. For systems with individual 5400 RPM or 7200 RPM drives, you might lower the value to 100, which represents an estimated proportion of the I/O operations per second (IOPS) available to older-generation disk drives that can perform about 100 IOPS.

Although you can specify a high value such as a million, in practice such large values have little benefit. Generally, a value higher than 20000 is not recommended unless you are certain that lower values are insufficient for your workload.

Consider write workload when tuning [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity). Systems with large write workloads are likely to benefit from a higher setting. A lower setting may be sufficient for systems with a small write workload.

The [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting is not a per buffer pool instance setting. Available I/O capacity is distributed equally among buffer pool instances for flushing activities.

You can set the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) value in the MySQL option file (**my.cnf** or **my.ini**) or modify it at runtime using a [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, which requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

#### Ignoring I/O Capacity at Checkpoints

The [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) variable, which is enabled by default, causes the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting to be ignored during bursts of I/O activity that occur at [checkpoints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint). To adhere to the I/O rate defined by the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting, disable [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync).

You can set the [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) value in the MySQL option file (**my.cnf** or **my.ini**) or modify it at runtime using a [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, which requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

#### Configuring an I/O Capacity Maximum

If flushing activity falls behind, **InnoDB** can flush more aggressively, at a higher rate of I/O operations per second (IOPS) than defined by the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) variable. The [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) variable defines a maximum number of IOPS performed by **InnoDB** background tasks in such situations.

If you specify an [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting at startup but do not specify a value for [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max), [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) defaults to twice the value of [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) or 2000, whichever value is greater.

When configuring [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max), twice the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) is often a good starting point. The default value of 2000 is intended for workloads that use an SSD or more than one regular disk drive. A setting of 2000 is likely too high for workloads that do not use SSDs or multiple disk drives, and could allow too much flushing. For a single regular disk drive, a setting between 200 and 400 is recommended. For a high-end, bus-attached SSD, consider a higher setting such as 2500. As with the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting, keep the setting as low as practical, but not so low that **InnoDB** cannot sufficiently extend rate of IOPS beyond the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting.

Consider write workload when tuning [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max). Systems with large write workloads may benefit from a higher setting. A lower setting may be sufficient for systems with a small write workload.

[**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) cannot be set to a value lower than the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) value.

Setting [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) to **DEFAULT** using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement (**SET GLOBAL innodb\_io\_capacity\_max=DEFAULT**) sets [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) to the maximum value.

The [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) limit applies to all buffer pool instances. It is not a per buffer pool instance setting.

### 15.8.8 Configuring Spin Lock Polling

**InnoDB** [mutexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mutex) and [rw-locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rw_lock) are typically reserved for short intervals. On a multi-core system, it can be more efficient for a thread to continuously check if it can acquire a mutex or rw-lock for a period of time before it sleeps. If the mutex or rw-lock becomes available during this period, the thread can continue immediately, in the same time slice. However, too-frequent polling of a shared object such as a mutex or rw-lock by multiple threads can cause “cache ping pong”, which results in processors invalidating portions of each other's cache. **InnoDB** minimizes this issue by forcing a random delay between polls to desychronize polling activity. The random delay is implemented as a spin-wait loop.

The duration of a spin-wait loop is determined by the number of PAUSE instructions that occur in the loop. That number is generated by randomly selecting an integer ranging from 0 up to but not including the [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) value, and multiplying that value by 50. (The multiplier value, 50, is hardcoded before MySQL 8.0.16, and configurable thereafter.) For example, an integer is randomly selected from the following range for an [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) setting of 6:

{0,1,2,3,4,5}

The selected integer is multiplied by 50, resulting in one of six possible PAUSE instruction values:

{0,50,100,150,200,250}

For that set of values, 250 is the maximum number of PAUSE instructions that can occur in a spin-wait loop. An [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) setting of 5 results in a set of five possible values **{0,50,100,150,200}**, where 200 is the maximum number of PAUSE instructions, and so on. In this way, the [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) setting controls the maximum delay between spin lock polls.

On a system where all processor cores share a fast cache memory, you might reduce the maximum delay or disable the busy loop altogether by setting [**innodb\_spin\_wait\_delay=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay). On a system with multiple processor chips, the effect of cache invalidation can be more significant and you might increase the maximum delay.

In the 100MHz Pentium era, an [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) unit was calibrated to be equivalent to one microsecond. That time equivalence did not hold, but PAUSE instruction duration remained fairly constant in terms of processor cycles relative to other CPU instructions until the introduction of the Skylake generation of processors, which have a comparatively longer PAUSE instruction. The [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) variable was introduced in MySQL 8.0.16 to provide a way to account for differences in PAUSE instruction duration.

The [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) variable controls the size of PAUSE instruction values. For example, assuming an [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) setting of 6, decreasing the [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) value from 50 (the default and previously hardcoded value) to 5 generates a set of smaller PAUSE instruction values:

{0,5,10,15,20,25}

The ability to increase or decrease PAUSE instruction values permits fine tuning **InnoDB** for different processor architectures. Smaller PAUSE instruction values would be appropriate for processor architectures with a comparatively longer PAUSE instruction, for example.

The [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) and [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) variables are dynamic. They can be specified in a MySQL option file or modified at runtime using a [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. Modifying the variables at runtime requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

### 15.8.9 Purge Configuration

**InnoDB** does not physically remove a row from the database immediately when you delete it with an SQL statement. A row and its index records are only physically removed when **InnoDB** discards the undo log record written for the deletion. This removal operation, which only occurs after the row is no longer required for multi-version concurrency control (MVCC) or rollback, is called a purge.

Purge runs on a periodic schedule. It parses and processes undo log pages from the history list, which is a list of undo log pages for committed transactions that is maintained by the **InnoDB** transaction system. Purge frees the undo log pages from the history list after processing them.

#### Configuring Purge Threads

Purge operations are performed in the background by one or more purge threads. The number of purge threads is controlled by the [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) variable. The default value is 4. If DML action is concentrated on a single table or a few tables, keep the setting low so that the threads do not contend with each other for access to the tables. If DML operations are spread across many tables, increase the setting. The maximum number of purge threads is 32.

The [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) setting is the maximum number of purge threads permitted. The purge system automatically adjusts the number of purge threads as necessary.

#### Configuring Purge Batch Size

The [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) variable defines the number of undo log pages that purge parses and processes in one batch from the history list. The default value is 300. In a multithreaded purge configuration, the coordinator purge thread divides [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) by [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) and assigns that number of pages to each purge thread.

The purge system also frees the undo log pages that are no longer required. It does so every 128 iterations through the undo logs. In addition to defining the number of undo log pages parsed and processed in a batch, the [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) variable defines the number of undo log pages that purge frees every 128 iterations through the undo logs.

The [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) variable is intended for advanced performance tuning and experimentation. Most users need not change [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) from its default value.

#### Configuring the Maximum Purge Lag

The [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) variable defines the desired maximum purge lag. When the purge lag exceeds the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) threshold, a delay is imposed on [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations to allow time for purge operations to catch up. The default value is 0, which means there is no maximum purge lag and no delay.

The **InnoDB** transaction system maintains a list of transactions that have index records delete-marked by [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations. The length of the list is the purge lag. Prior to MySQL 8.0.14, the purge lag delay is calculated by the following formula, which results in a minimum delay of 5000 microseconds:

(purge lag/innodb\_max\_purge\_lag - 0.5) \* 10000

As of MySQL 8.0.14, the purge lag delay is calculated by the following revised formula, which reduces the minimum delay to 5 microseconds. A delay of 5 microseconds is more appropriate for modern systems.

(purge\_lag/innodb\_max\_purge\_lag - 0.9995) \* 10000

The delay is calculated at the beginning of a purge batch.

A typical [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) setting for a problematic workload might be 1000000 (1 million), assuming that transactions are small, only 100 bytes in size, and it is permissible to have 100MB of unpurged table rows.

The purge lag is presented as the **History list length** value in the **TRANSACTIONS** section of [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output.

mysql> SHOW ENGINE INNODB STATUS;

...

------------

TRANSACTIONS

------------

Trx id counter 0 290328385

Purge done for trx's n:o < 0 290315608 undo n:o < 0 17

History list length 20

The **History list length** is typically a low value, usually less than a few thousand, but a write-heavy workload or long running transactions can cause it to increase, even for transactions that are read only. The reason that a long running transaction can cause the **History list length** to increase is that under a consistent read transaction isolation level such as **REPEATABLE READ**, a transaction must return the same result as when the read view for that transaction was created. Consequently, the **InnoDB** multi-version concurrency control (MVCC) system must keep a copy of the data in the undo log until all transactions that depend on that data have completed. The following are examples of long running transactions that could cause the **History list length** to increase:

A [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) operation that uses the [--single-transaction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_single-transaction) option while there is a significant amount of concurrent DML.

Running a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) query after disabling [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), and forgetting to issue an explicit **COMMIT** or **ROLLBACK**.

To prevent excessive delays in extreme situations where the purge lag becomes huge, you can limit the delay by setting the [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) variable. The [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) variable specifies the maximum delay in microseconds for the delay imposed when the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) threshold is exceeded. The specified [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) value is an upper limit on the delay period calculated by the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) formula.

#### Purge and Undo Tablespace Truncation

The purge system is also responsible for truncating undo tablespaces. You can configure the [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) variable to control the frequency with which the purge system looks for undo tablespaces to truncate. For more information, see [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

### 15.8.10 Configuring Optimizer Statistics for InnoDB

[15.8.10.1 Configuring Persistent Optimizer Statistics Parameters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats)

[15.8.10.2 Configuring Non-Persistent Optimizer Statistics Parameters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation)

[15.8.10.3 Estimating ANALYZE TABLE Complexity for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-analyze-table-complexity)

This section describes how to configure persistent and non-persistent optimizer statistics for **InnoDB** tables.

Persistent optimizer statistics are persisted across server restarts, allowing for greater [plan stability](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_plan_stability) and more consistent query performance. Persistent optimizer statistics also provide control and flexibility with these additional benefits:

You can use the [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) configuration option to control whether statistics are updated automatically after substantial changes to a table.

You can use the **STATS\_PERSISTENT**, **STATS\_AUTO\_RECALC**, and **STATS\_SAMPLE\_PAGES** clauses with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements to configure optimizer statistics for individual tables.

You can query optimizer statistics data in the **mysql.innodb\_table\_stats** and **mysql.innodb\_index\_stats** tables.

You can view the **last\_update** column of the **mysql.innodb\_table\_stats** and **mysql.innodb\_index\_stats** tables to see when statistics were last updated.

You can manually modify the **mysql.innodb\_table\_stats** and **mysql.innodb\_index\_stats** tables to force a specific query optimization plan or to test alternative plans without modifying the database.

The persistent optimizer statistics feature is enabled by default ([**innodb\_stats\_persistent=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent)).

Non-persistent optimizer statistics are cleared on each server restart and after some other operations, and recomputed on the next table access. As a result, different estimates could be produced when recomputing statistics, leading to different choices in execution plans and variations in query performance.

This section also provides information about estimating [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) complexity, which may be useful when attempting to achieve a balance between accurate statistics and [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) execution time.

#### 15.8.10.1 Configuring Persistent Optimizer Statistics Parameters

The persistent optimizer statistics feature improves [plan stability](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_plan_stability) by storing statistics to disk and making them persistent across server restarts so that the [optimizer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_optimizer) is more likely to make consistent choices each time for a given query.

Optimizer statistics are persisted to disk when [**innodb\_stats\_persistent=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) or when individual tables are defined with [**STATS\_PERSISTENT=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is enabled by default.

Formerly, optimizer statistics were cleared when restarting the server and after some other types of operations, and recomputed on the next table access. Consequently, different estimates could be produced when recalculating statistics leading to different choices in query execution plans and variation in query performance.

Persistent statistics are stored in the **mysql.innodb\_table\_stats** and **mysql.innodb\_index\_stats** tables. See [Section 15.8.10.1.5, “InnoDB Persistent Statistics Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats-tables).

If you prefer not to persist optimizer statistics to disk, see [Section 15.8.10.2, “Configuring Non-Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation)

##### 15.8.10.1.1 Configuring Automatic Statistics Calculation for Persistent Optimizer Statistics

The [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) variable, which is enabled by default, controls whether statistics are calculated automatically when a table undergoes changes to more than 10% of its rows. You can also configure automatic statistics recalculation for individual tables by specifying the **STATS\_AUTO\_RECALC** clause when creating or altering a table.

Because of the asynchronous nature of automatic statistics recalculation, which occurs in the background, statistics may not be recalculated instantly after running a DML operation that affects more than 10% of a table, even when [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) is enabled. Statistics recalculation can be delayed by few seconds in some cases. If up-to-date statistics are required immediately, run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to initiate a synchronous (foreground) recalculation of statistics.

If [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) is disabled, you can ensure the accuracy of optimizer statistics by executing the [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) statement after making substantial changes to indexed columns. You might also consider adding [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to setup scripts that you run after loading data, and running [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) on a schedule at times of low activity.

When an index is added to an existing table, or when a column is added or dropped, index statistics are calculated and added to the **innodb\_index\_stats** table regardless of the value of [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc).

##### 15.8.10.1.2 Configuring Optimizer Statistics Parameters for Individual Tables

[**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent), [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc), and [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) are global variables. To override these system-wide settings and configure optimizer statistics parameters for individual tables, you can define **STATS\_PERSISTENT**, **STATS\_AUTO\_RECALC**, and **STATS\_SAMPLE\_PAGES** clauses in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements.

**STATS\_PERSISTENT** specifies whether to enable [persistent statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_persistent_statistics) for an **InnoDB** table. The value **DEFAULT** causes the persistent statistics setting for the table to be determined by the [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) setting. A value of **1** enables persistent statistics for the table, while a value of **0** disables the feature. After enabling persistent statistics for an individual table, use [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to calculate statistics after table data is loaded.

**STATS\_AUTO\_RECALC** specifies whether to automatically recalculate [persistent statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_persistent_statistics). The value **DEFAULT** causes the persistent statistics setting for the table to be determined by the [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) setting. A value of **1** causes statistics to be recalculated when 10% of table data has changed. A value **0** prevents automatic recalculation for the table. When using a value of 0, use [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to recalculate statistics after making substantial changes to the table.

**STATS\_SAMPLE\_PAGES** specifies the number of index pages to sample when cardinality and other statistics are calculated for an indexed column, by an [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) operation, for example.

All three clauses are specified in the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) example:

CREATE TABLE `t1` (

`id` int(8) NOT NULL auto\_increment,

`data` varchar(255),

`date` datetime,

PRIMARY KEY (`id`),

INDEX `DATE\_IX` (`date`)

) ENGINE=InnoDB,

STATS\_PERSISTENT=1,

STATS\_AUTO\_RECALC=1,

STATS\_SAMPLE\_PAGES=25;

##### 15.8.10.1.3 Configuring the Number of Sampled Pages for InnoDB Optimizer Statistics

The optimizer uses estimated [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) about key distributions to choose the indexes for an execution plan, based on the relative [selectivity](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_selectivity) of the index. Operations such as [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) cause **InnoDB** to sample random pages from each index on a table to estimate the [cardinality](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cardinality) of the index. This sampling technique is known as a [random dive](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_random_dive).

The [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) controls the number of sampled pages. You can adjust the setting at runtime to manage the quality of statistics estimates used by the optimizer. The default value is 20. Consider modifying the setting when encountering the following issues:

Statistics are not accurate enough and the optimizer chooses suboptimal plans, as shown in [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output. You can check the accuracy of statistics by comparing the actual cardinality of an index (determined by running [**SELECT DISTINCT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) on the index columns) with the estimates in the **mysql.innodb\_index\_stats** table.

If it is determined that statistics are not accurate enough, the value of [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) should be increased until the statistics estimates are sufficiently accurate. Increasing [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) too much, however, could cause [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to run slowly.

[***ANALYZE TABLE***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) is too slow. In this case [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) should be decreased until [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) execution time is acceptable. Decreasing the value too much, however, could lead to the first problem of inaccurate statistics and suboptimal query execution plans.

If a balance cannot be achieved between accurate statistics and [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) execution time, consider decreasing the number of indexed columns in the table or limiting the number of partitions to reduce [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) complexity. The number of columns in the table's primary key is also important to consider, as primary key columns are appended to each nonunique index.

For related information, see [Section 15.8.10.3, “Estimating ANALYZE TABLE Complexity for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-analyze-table-complexity).

##### 15.8.10.1.4 Including Delete-marked Records in Persistent Statistics Calculations

By default, **InnoDB** reads uncommitted data when calculating statistics. In the case of an uncommitted transaction that deletes rows from a table, delete-marked records are excluded when calculating row estimates and index statistics, which can lead to non-optimal execution plans for other transactions that are operating on the table concurrently using a transaction isolation level other than [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted). To avoid this scenario, [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) can be enabled to ensure that delete-marked records are included when calculating persistent optimizer statistics.

When [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) is enabled, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) considers delete-marked records when recalculating statistics.

[**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) is a global setting that affects all **InnoDB** tables, and it is only applicable to persistent optimizer statistics.

##### 15.8.10.1.5 InnoDB Persistent Statistics Tables

The persistent statistics feature relies on the internally managed tables in the **mysql** database, named **innodb\_table\_stats** and **innodb\_index\_stats**. These tables are set up automatically in all install, upgrade, and build-from-source procedures.

**Table 15.6 Columns of innodb\_table\_stats**

| **Column name** | **Description** |
| --- | --- |
| **database\_name** | Database name |
| **table\_name** | Table name, partition name, or subpartition name |
| **last\_update** | A timestamp indicating the last time that **InnoDB** updated this row |
| **n\_rows** | The number of rows in the table |
| **clustered\_index\_size** | The size of the primary index, in pages |
| **sum\_of\_other\_index\_sizes** | The total size of other (non-primary) indexes, in pages |

**Table 15.7 Columns of innodb\_index\_stats**

| **Column name** | **Description** |
| --- | --- |
| **database\_name** | Database name |
| **table\_name** | Table name, partition name, or subpartition name |
| **index\_name** | Index name |
| **last\_update** | A timestamp indicating the last time the row was updated |
| **stat\_name** | The name of the statistic, whose value is reported in the **stat\_value** column |
| **stat\_value** | The value of the statistic that is named in **stat\_name** column |
| **sample\_size** | The number of pages sampled for the estimate provided in the **stat\_value** column |
| **stat\_description** | Description of the statistic that is named in the **stat\_name** column |

The **innodb\_table\_stats** and **innodb\_index\_stats** tables include a **last\_update** column that shows when index statistics were last updated:

mysql> **SELECT \* FROM innodb\_table\_stats \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

database\_name: sakila

table\_name: actor

last\_update: 2014-05-28 16:16:44

n\_rows: 200

clustered\_index\_size: 1

sum\_of\_other\_index\_sizes: 1

...

mysql> **SELECT \* FROM innodb\_index\_stats \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

database\_name: sakila

table\_name: actor

index\_name: PRIMARY

last\_update: 2014-05-28 16:16:44

stat\_name: n\_diff\_pfx01

stat\_value: 200

sample\_size: 1

...

The **innodb\_table\_stats** and **innodb\_index\_stats** tables can be updated manually, which makes it possible to force a specific query optimization plan or test alternative plans without modifying the database. If you manually update statistics, use the **FLUSH TABLE *tbl\_name*** statement to load the updated statistics.

Persistent statistics are considered local information, because they relate to the server instance. The **innodb\_table\_stats** and **innodb\_index\_stats** tables are therefore not replicated when automatic statistics recalculation takes place. If you run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to initiate a synchronous recalculation of statistics, the statement is replicated (unless you suppressed logging for it), and recalculation takes place on replicas.

##### 15.8.10.1.6 InnoDB Persistent Statistics Tables Example

The **innodb\_table\_stats** table contains one row for each table. The following example demonstrates the type of data collected.

Table **t1** contains a primary index (columns **a**, **b**) secondary index (columns **c**, **d**), and unique index (columns **e**,**f**):

CREATE TABLE t1 (

a INT, b INT, c INT, d INT, e INT, f INT,

PRIMARY KEY (a, b), KEY i1 (c, d), UNIQUE KEY i2uniq (e, f)

) ENGINE=INNODB;

After inserting five rows of sample data, table **t1** appears as follows:

mysql> **SELECT \* FROM t1;**

+---+---+------+------+------+------+

| a | b | c | d | e | f |

+---+---+------+------+------+------+

| 1 | 1 | 10 | 11 | 100 | 101 |

| 1 | 2 | 10 | 11 | 200 | 102 |

| 1 | 3 | 10 | 11 | 100 | 103 |

| 1 | 4 | 10 | 12 | 200 | 104 |

| 1 | 5 | 10 | 12 | 100 | 105 |

+---+---+------+------+------+------+

To immediately update statistics, run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) (if [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) is enabled, statistics are updated automatically within a few seconds assuming that the 10% threshold for changed table rows is reached):

mysql> **ANALYZE TABLE t1;**

+---------+---------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+---------+---------+----------+----------+

| test.t1 | analyze | status | OK |

+---------+---------+----------+----------+

Table statistics for table **t1** show the last time **InnoDB** updated the table statistics (**2014-03-14 14:36:34**), the number of rows in the table (**5**), the clustered index size (**1** page), and the combined size of the other indexes (**2** pages).

mysql> **SELECT \* FROM mysql.innodb\_table\_stats WHERE table\_name like 't1'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

database\_name: test

table\_name: t1

last\_update: 2014-03-14 14:36:34

n\_rows: 5

clustered\_index\_size: 1

sum\_of\_other\_index\_sizes: 2

The **innodb\_index\_stats** table contains multiple rows for each index. Each row in the **innodb\_index\_stats** table provides data related to a particular index statistic which is named in the **stat\_name** column and described in the **stat\_description** column. For example:

mysql> **SELECT index\_name, stat\_name, stat\_value, stat\_description**

**FROM mysql.innodb\_index\_stats WHERE table\_name like 't1';**

+------------+--------------+------------+-----------------------------------+

| index\_name | stat\_name | stat\_value | stat\_description |

+------------+--------------+------------+-----------------------------------+

| PRIMARY | n\_diff\_pfx01 | 1 | a |

| PRIMARY | n\_diff\_pfx02 | 5 | a,b |

| PRIMARY | n\_leaf\_pages | 1 | Number of leaf pages in the index |

| PRIMARY | size | 1 | Number of pages in the index |

| i1 | n\_diff\_pfx01 | 1 | c |

| i1 | n\_diff\_pfx02 | 2 | c,d |

| i1 | n\_diff\_pfx03 | 2 | c,d,a |

| i1 | n\_diff\_pfx04 | 5 | c,d,a,b |

| i1 | n\_leaf\_pages | 1 | Number of leaf pages in the index |

| i1 | size | 1 | Number of pages in the index |

| i2uniq | n\_diff\_pfx01 | 2 | e |

| i2uniq | n\_diff\_pfx02 | 5 | e,f |

| i2uniq | n\_leaf\_pages | 1 | Number of leaf pages in the index |

| i2uniq | size | 1 | Number of pages in the index |

+------------+--------------+------------+-----------------------------------+

The **stat\_name** column shows the following types of statistics:

**size**: Where **stat\_name**=**size**, the **stat\_value** column displays the total number of pages in the index.

**n\_leaf\_pages**: Where **stat\_name**=**n\_leaf\_pages**, the **stat\_value** column displays the number of leaf pages in the index.

**n\_diff\_pfx*NN***: Where **stat\_name**=**n\_diff\_pfx01**, the **stat\_value** column displays the number of distinct values in the first column of the index. Where **stat\_name**=**n\_diff\_pfx02**, the **stat\_value** column displays the number of distinct values in the first two columns of the index, and so on. Where **stat\_name**=**n\_diff\_pfx*NN***, the **stat\_description** column shows a comma separated list of the index columns that are counted.

To further illustrate the **n\_diff\_pfx*NN*** statistic, which provides cardinality data, consider once again the **t1** table example that was introduced previously. As shown below, the **t1** table is created with a primary index (columns **a**, **b**), a secondary index (columns **c**, **d**), and a unique index (columns **e**, **f**):

CREATE TABLE t1 (

a INT, b INT, c INT, d INT, e INT, f INT,

PRIMARY KEY (a, b), KEY i1 (c, d), UNIQUE KEY i2uniq (e, f)

) ENGINE=INNODB;

After inserting five rows of sample data, table **t1** appears as follows:

mysql> **SELECT \* FROM t1;**

+---+---+------+------+------+------+

| a | b | c | d | e | f |

+---+---+------+------+------+------+

| 1 | 1 | 10 | 11 | 100 | 101 |

| 1 | 2 | 10 | 11 | 200 | 102 |

| 1 | 3 | 10 | 11 | 100 | 103 |

| 1 | 4 | 10 | 12 | 200 | 104 |

| 1 | 5 | 10 | 12 | 100 | 105 |

+---+---+------+------+------+------+

When you query the **index\_name**, **stat\_name**, **stat\_value**, and **stat\_description**, where **stat\_name LIKE 'n\_diff%'**, the following result set is returned:

mysql> **SELECT index\_name, stat\_name, stat\_value, stat\_description**

**FROM mysql.innodb\_index\_stats**

**WHERE table\_name like 't1' AND stat\_name LIKE 'n\_diff%';**

+------------+--------------+------------+------------------+

| index\_name | stat\_name | stat\_value | stat\_description |

+------------+--------------+------------+------------------+

| PRIMARY | n\_diff\_pfx01 | 1 | a |

| PRIMARY | n\_diff\_pfx02 | 5 | a,b |

| i1 | n\_diff\_pfx01 | 1 | c |

| i1 | n\_diff\_pfx02 | 2 | c,d |

| i1 | n\_diff\_pfx03 | 2 | c,d,a |

| i1 | n\_diff\_pfx04 | 5 | c,d,a,b |

| i2uniq | n\_diff\_pfx01 | 2 | e |

| i2uniq | n\_diff\_pfx02 | 5 | e,f |

+------------+--------------+------------+------------------+

For the **PRIMARY** index, there are two **n\_diff%** rows. The number of rows is equal to the number of columns in the index.

**Note**

For nonunique indexes, **InnoDB** appends the columns of the primary key.

Where **index\_name**=**PRIMARY** and **stat\_name**=**n\_diff\_pfx01**, the **stat\_value** is **1**, which indicates that there is a single distinct value in the first column of the index (column **a**). The number of distinct values in column **a** is confirmed by viewing the data in column **a** in table **t1**, in which there is a single distinct value (**1**). The counted column (**a**) is shown in the **stat\_description** column of the result set.

Where **index\_name**=**PRIMARY** and **stat\_name**=**n\_diff\_pfx02**, the **stat\_value** is **5**, which indicates that there are five distinct values in the two columns of the index (**a,b**). The number of distinct values in columns **a** and **b** is confirmed by viewing the data in columns **a** and **b** in table **t1**, in which there are five distinct values: (**1,1**), (**1,2**), (**1,3**), (**1,4**) and (**1,5**). The counted columns (**a,b**) are shown in the **stat\_description** column of the result set.

For the secondary index (**i1**), there are four **n\_diff%** rows. Only two columns are defined for the secondary index (**c,d**) but there are four **n\_diff%** rows for the secondary index because **InnoDB** suffixes all nonunique indexes with the primary key. As a result, there are four **n\_diff%** rows instead of two to account for the both the secondary index columns (**c,d**) and the primary key columns (**a,b**).

Where **index\_name**=**i1** and **stat\_name**=**n\_diff\_pfx01**, the **stat\_value** is **1**, which indicates that there is a single distinct value in the first column of the index (column **c**). The number of distinct values in column **c** is confirmed by viewing the data in column **c** in table **t1**, in which there is a single distinct value: (**10**). The counted column (**c**) is shown in the **stat\_description** column of the result set.

Where **index\_name**=**i1** and **stat\_name**=**n\_diff\_pfx02**, the **stat\_value** is **2**, which indicates that there are two distinct values in the first two columns of the index (**c,d**). The number of distinct values in columns **c** an **d** is confirmed by viewing the data in columns **c** and **d** in table **t1**, in which there are two distinct values: (**10,11**) and (**10,12**). The counted columns (**c,d**) are shown in the **stat\_description** column of the result set.

Where **index\_name**=**i1** and **stat\_name**=**n\_diff\_pfx03**, the **stat\_value** is **2**, which indicates that there are two distinct values in the first three columns of the index (**c,d,a**). The number of distinct values in columns **c**, **d**, and **a** is confirmed by viewing the data in column **c**, **d**, and **a** in table **t1**, in which there are two distinct values: (**10,11,1**) and (**10,12,1**). The counted columns (**c,d,a**) are shown in the **stat\_description** column of the result set.

Where **index\_name**=**i1** and **stat\_name**=**n\_diff\_pfx04**, the **stat\_value** is **5**, which indicates that there are five distinct values in the four columns of the index (**c,d,a,b**). The number of distinct values in columns **c**, **d**, **a** and **b** is confirmed by viewing the data in columns **c**, **d**, **a**, and **b** in table **t1**, in which there are five distinct values: (**10,11,1,1**), (**10,11,1,2**), (**10,11,1,3**), (**10,12,1,4**), and (**10,12,1,5**). The counted columns (**c,d,a,b**) are shown in the **stat\_description** column of the result set.

For the unique index (**i2uniq**), there are two **n\_diff%** rows.

Where **index\_name**=**i2uniq** and **stat\_name**=**n\_diff\_pfx01**, the **stat\_value** is **2**, which indicates that there are two distinct values in the first column of the index (column **e**). The number of distinct values in column **e** is confirmed by viewing the data in column **e** in table **t1**, in which there are two distinct values: (**100**) and (**200**). The counted column (**e**) is shown in the **stat\_description** column of the result set.

Where **index\_name**=**i2uniq** and **stat\_name**=**n\_diff\_pfx02**, the **stat\_value** is **5**, which indicates that there are five distinct values in the two columns of the index (**e,f**). The number of distinct values in columns **e** and **f** is confirmed by viewing the data in columns **e** and **f** in table **t1**, in which there are five distinct values: (**100,101**), (**200,102**), (**100,103**), (**200,104**), and (**100,105**). The counted columns (**e,f**) are shown in the **stat\_description** column of the result set.

##### 15.8.10.1.7 Retrieving Index Size Using the innodb\_index\_stats Table

You can retrieve the index size for tables, partitions, or subpartitions can using the **innodb\_index\_stats** table. In the following example, index sizes are retrieved for table **t1**. For a definition of table **t1** and corresponding index statistics, see [Section 15.8.10.1.6, “InnoDB Persistent Statistics Tables Example”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats-tables-example).

mysql> **SELECT SUM(stat\_value) pages, index\_name,**

**SUM(stat\_value)\*@@innodb\_page\_size size**

**FROM mysql.innodb\_index\_stats WHERE table\_name='t1'**

**AND stat\_name = 'size' GROUP BY index\_name;**

+-------+------------+-------+

| pages | index\_name | size |

+-------+------------+-------+

| 1 | PRIMARY | 16384 |

| 1 | i1 | 16384 |

| 1 | i2uniq | 16384 |

+-------+------------+-------+

For partitions or subpartitions, you can use the same query with a modified **WHERE** clause to retrieve index sizes. For example, the following query retrieves index sizes for partitions of table **t1**:

mysql> **SELECT SUM(stat\_value) pages, index\_name,**

**SUM(stat\_value)\*@@innodb\_page\_size size**

**FROM mysql.innodb\_index\_stats WHERE table\_name like 't1#P%'**

**AND stat\_name = 'size' GROUP BY index\_name;**

#### 15.8.10.2 Configuring Non-Persistent Optimizer Statistics Parameters

This section describes how to configure non-persistent optimizer statistics. Optimizer statistics are not persisted to disk when [**innodb\_stats\_persistent=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) or when individual tables are created or altered with [**STATS\_PERSISTENT=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). Instead, statistics are stored in memory, and are lost when the server is shut down. Statistics are also updated periodically by certain operations and under certain conditions.

Optimizer statistics are persisted to disk by default, enabled by the [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) configuration option. For information about persistent optimizer statistics, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

##### Optimizer Statistics Updates

Non-persistent optimizer statistics are updated when:

Running [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table).

Running [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status), [**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index), or querying the [**INFORMATION\_SCHEMA.TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) or [**INFORMATION\_SCHEMA.STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table) tables with the [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) option enabled.

The default setting for [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) is **OFF**. Enabling [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) may reduce access speed for schemas that have a large number of tables or indexes, and reduce stability of execution plans for queries that involve **InnoDB** tables. [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) is configured globally using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) statement.

SET GLOBAL innodb\_stats\_on\_metadata=ON

**Note**

[**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) only applies when optimizer [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) are configured to be non-persistent (when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is disabled).

Starting a [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client with the [--auto-rehash](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_auto-rehash) option enabled, which is the default. The [auto-rehash](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_auto-rehash) option causes all **InnoDB** tables to be opened, and the open table operations cause statistics to be recalculated.

To improve the start up time of the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client and to updating statistics, you can turn off [auto-rehash](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_auto-rehash) using the [--disable-auto-rehash](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_auto-rehash) option. The [auto-rehash](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_auto-rehash) feature enables automatic name completion of database, table, and column names for interactive users.

A table is first opened.

**InnoDB** detects that 1 / 16 of table has been modified since the last time statistics were updated.

##### Configuring the Number of Sampled Pages

The MySQL query optimizer uses estimated [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) about key distributions to choose the indexes for an execution plan, based on the relative [selectivity](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_selectivity) of the index. When **InnoDB** updates optimizer statistics, it samples random pages from each index on a table to estimate the [cardinality](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cardinality) of the index. (This technique is known as [random dives](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_random_dive).)

To give you control over the quality of the statistics estimate (and thus better information for the query optimizer), you can change the number of sampled pages using the parameter [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages). The default number of sampled pages is 8, which could be insufficient to produce an accurate estimate, leading to poor index choices by the query optimizer. This technique is especially important for large tables and tables used in [joins](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_join). Unnecessary [full table scans](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_full_table_scan) for such tables can be a substantial performance issue. See [Section 8.2.1.23, “Avoiding Full Table Scans”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#table-scan-avoidance) for tips on tuning such queries. [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) is a global parameter that can be set at runtime.

The value of [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) affects the index sampling for all **InnoDB** tables and indexes when [**innodb\_stats\_persistent=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent). Be aware of the following potentially significant impacts when you change the index sample size:

Small values like 1 or 2 can result in inaccurate estimates of cardinality.

Increasing the [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) value might require more disk reads. Values much larger than 8 (say, 100), can cause a significant slowdown in the time it takes to open a table or execute **SHOW TABLE STATUS**.

The optimizer might choose very different query plans based on different estimates of index selectivity.

Whatever value of [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) works best for a system, set the option and leave it at that value. Choose a value that results in reasonably accurate estimates for all tables in your database without requiring excessive I/O. Because the statistics are automatically recalculated at various times other than on execution of [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), it does not make sense to increase the index sample size, run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), then decrease sample size again.

Smaller tables generally require fewer index samples than larger tables. If your database has many large tables, consider using a higher value for [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) than if you have mostly smaller tables.

#### 15.8.10.3 Estimating ANALYZE TABLE Complexity for InnoDB Tables

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) complexity for **InnoDB** tables is dependent on:

The number of pages sampled, as defined by [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages).

The number of indexed columns in a table

The number of partitions. If a table has no partitions, the number of partitions is considered to be 1.

Using these parameters, an approximate formula for estimating [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) complexity would be:

The value of [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) \* number of indexed columns in a table \* the number of partitions

Typically, the greater the resulting value, the greater the execution time for [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table).

**Note**

[**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) defines the number of pages sampled at a global level. To set the number of pages sampled for an individual table, use the **STATS\_SAMPLE\_PAGES** option with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

If [**innodb\_stats\_persistent=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent), the number of pages sampled is defined by [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages). See [Section 15.8.10.2, “Configuring Non-Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation) for additional information.

For a more in-depth approach to estimating **ANALYZE TABLE** complexity, consider the following example.

In [Big O notation](http://en.wikipedia.org/wiki/Big_O_notation), [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) complexity is described as:

O(n\_sample

\* (n\_cols\_in\_uniq\_i

+ n\_cols\_in\_non\_uniq\_i

+ n\_cols\_in\_pk \* (1 + n\_non\_uniq\_i))

\* n\_part)

where:

**n\_sample** is the number of pages sampled (defined by [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages))

**n\_cols\_in\_uniq\_i** is total number of all columns in all unique indexes (not counting the primary key columns)

**n\_cols\_in\_non\_uniq\_i** is the total number of all columns in all nonunique indexes

**n\_cols\_in\_pk** is the number of columns in the primary key (if a primary key is not defined, **InnoDB** creates a single column primary key internally)

**n\_non\_uniq\_i** is the number of nonunique indexes in the table

**n\_part** is the number of partitions. If no partitions are defined, the table is considered to be a single partition.

Now, consider the following table (table **t**), which has a primary key (2 columns), a unique index (2 columns), and two nonunique indexes (two columns each):

CREATE TABLE t (

a INT,

b INT,

c INT,

d INT,

e INT,

f INT,

g INT,

h INT,

PRIMARY KEY (a, b),

UNIQUE KEY i1uniq (c, d),

KEY i2nonuniq (e, f),

KEY i3nonuniq (g, h)

);

For the column and index data required by the algorithm described above, query the **mysql.innodb\_index\_stats** persistent index statistics table for table **t**. The **n\_diff\_pfx%** statistics show the columns that are counted for each index. For example, columns **a** and **b** are counted for the primary key index. For the nonunique indexes, the primary key columns (a,b) are counted in addition to the user defined columns.

**Note**

For additional information about the **InnoDB** persistent statistics tables, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats)

mysql> **SELECT index\_name, stat\_name, stat\_description**

**FROM mysql.innodb\_index\_stats WHERE**

**database\_name='test' AND**

**table\_name='t' AND**

**stat\_name like 'n\_diff\_pfx%';**

+------------+--------------+------------------+

| index\_name | stat\_name | stat\_description |

+------------+--------------+------------------+

| PRIMARY | n\_diff\_pfx01 | a |

| PRIMARY | n\_diff\_pfx02 | a,b |

| i1uniq | n\_diff\_pfx01 | c |

| i1uniq | n\_diff\_pfx02 | c,d |

| i2nonuniq | n\_diff\_pfx01 | e |

| i2nonuniq | n\_diff\_pfx02 | e,f |

| i2nonuniq | n\_diff\_pfx03 | e,f,a |

| i2nonuniq | n\_diff\_pfx04 | e,f,a,b |

| i3nonuniq | n\_diff\_pfx01 | g |

| i3nonuniq | n\_diff\_pfx02 | g,h |

| i3nonuniq | n\_diff\_pfx03 | g,h,a |

| i3nonuniq | n\_diff\_pfx04 | g,h,a,b |

+------------+--------------+------------------+

Based on the index statistics data shown above and the table definition, the following values can be determined:

**n\_cols\_in\_uniq\_i**, the total number of all columns in all unique indexes not counting the primary key columns, is 2 (**c** and **d**)

**n\_cols\_in\_non\_uniq\_i**, the total number of all columns in all nonunique indexes, is 4 (**e**, **f**, **g** and **h**)

**n\_cols\_in\_pk**, the number of columns in the primary key, is 2 (**a** and **b**)

**n\_non\_uniq\_i**, the number of nonunique indexes in the table, is 2 (**i2nonuniq** and **i3nonuniq**))

**n\_part**, the number of partitions, is 1.

You can now calculate **innodb\_stats\_persistent\_sample\_pages** \* (2 + 4 + 2 \* (1 + 2)) \* 1 to determine the number of leaf pages that are scanned. With **innodb\_stats\_persistent\_sample\_pages** set to the default value of **20**, and with a default page size of 16 **KiB** ([**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size)=16384), you can then estimate that 20 \* 12 \* 16384 **bytes** are read for table **t**, or about 4 **MiB**.

**Note**

All 4 **MiB** may not be read from disk, as some leaf pages may already be cached in the buffer pool.

### 15.8.11 Configuring the Merge Threshold for Index Pages

You can configure the **MERGE\_THRESHOLD** value for index pages. If the “page-full” percentage for an index page falls below the **MERGE\_THRESHOLD** value when a row is deleted or when a row is shortened by an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operation, **InnoDB** attempts to merge the index page with a neighboring index page. The default **MERGE\_THRESHOLD** value is 50, which is the previously hardcoded value. The minimum **MERGE\_THRESHOLD** value is 1 and the maximum value is 50.

When the “page-full” percentage for an index page falls below 50%, which is the default **MERGE\_THRESHOLD** setting, **InnoDB** attempts to merge the index page with a neighboring page. If both pages are close to 50% full, a page split can occur soon after the pages are merged. If this merge-split behavior occurs frequently, it can have an adverse affect on performance. To avoid frequent merge-splits, you can lower the **MERGE\_THRESHOLD** value so that **InnoDB** attempts page merges at a lower “page-full” percentage. Merging pages at a lower page-full percentage leaves more room in index pages and helps reduce merge-split behavior.

The **MERGE\_THRESHOLD** for index pages can be defined for a table or for individual indexes. A **MERGE\_THRESHOLD** value defined for an individual index takes priority over a **MERGE\_THRESHOLD** value defined for the table. If undefined, the **MERGE\_THRESHOLD** value defaults to 50.

#### Setting MERGE\_THRESHOLD for a Table

You can set the **MERGE\_THRESHOLD** value for a table using the ***table\_option*** **COMMENT** clause of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. For example:

CREATE TABLE t1 (

id INT,

KEY id\_index (id)

) COMMENT='MERGE\_THRESHOLD=45';

You can also set the **MERGE\_THRESHOLD** value for an existing table using the ***table\_option*** **COMMENT** clause with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

CREATE TABLE t1 (

id INT,

KEY id\_index (id)

);

ALTER TABLE t1 COMMENT='MERGE\_THRESHOLD=40';

#### Setting MERGE\_THRESHOLD for Individual Indexes

To set the **MERGE\_THRESHOLD** value for an individual index, you can use the ***index\_option*** **COMMENT** clause with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), as shown in the following examples:

Setting **MERGE\_THRESHOLD** for an individual index using [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table):

CREATE TABLE t1 (

id INT,

KEY id\_index (id) COMMENT 'MERGE\_THRESHOLD=40'

);

Setting **MERGE\_THRESHOLD** for an individual index using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

CREATE TABLE t1 (

id INT,

KEY id\_index (id)

);

ALTER TABLE t1 DROP KEY id\_index;

ALTER TABLE t1 ADD KEY id\_index (id) COMMENT 'MERGE\_THRESHOLD=40';

Setting **MERGE\_THRESHOLD** for an individual index using [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index):

CREATE TABLE t1 (id INT);

CREATE INDEX id\_index ON t1 (id) COMMENT 'MERGE\_THRESHOLD=40';

**Note**

You cannot modify the **MERGE\_THRESHOLD** value at the index level for **GEN\_CLUST\_INDEX**, which is the clustered index created by **InnoDB** when an **InnoDB** table is created without a primary key or unique key index. You can only modify the **MERGE\_THRESHOLD** value for **GEN\_CLUST\_INDEX** by setting **MERGE\_THRESHOLD** for the table.

#### Querying the MERGE\_THRESHOLD Value for an Index

The current **MERGE\_THRESHOLD** value for an index can be obtained by querying the [**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) table. For example:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_INDEXES WHERE NAME='id\_index' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INDEX\_ID: 91

NAME: id\_index

TABLE\_ID: 68

TYPE: 0

N\_FIELDS: 1

PAGE\_NO: 4

SPACE: 57

MERGE\_THRESHOLD: 40

You can use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) to view the **MERGE\_THRESHOLD** value for a table, if explicitly defined using the ***table\_option*** **COMMENT** clause:

mysql> **SHOW CREATE TABLE t2 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t2

Create Table: CREATE TABLE `t2` (

`id` int(11) DEFAULT NULL,

KEY `id\_index` (`id`) COMMENT 'MERGE\_THRESHOLD=40'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4

**Note**

A **MERGE\_THRESHOLD** value defined at the index level takes priority over a **MERGE\_THRESHOLD** value defined for the table. If undefined, **MERGE\_THRESHOLD** defaults to 50% (**MERGE\_THRESHOLD=50**, which is the previously hardcoded value.

Likewise, you can use [**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) to view the **MERGE\_THRESHOLD** value for an index, if explicitly defined using the ***index\_option*** **COMMENT** clause:

mysql> **SHOW INDEX FROM t2 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t2

Non\_unique: 1

Key\_name: id\_index

Seq\_in\_index: 1

Column\_name: id

Collation: A

Cardinality: 0

Sub\_part: NULL

Packed: NULL

Null: YES

Index\_type: BTREE

Comment:

Index\_comment: MERGE\_THRESHOLD=40

#### Measuring the Effect of MERGE\_THRESHOLD Settings

The [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table provides two counters that can be used to measure the effect of a **MERGE\_THRESHOLD** setting on index page merges.

mysql> **SELECT NAME, COMMENT FROM INFORMATION\_SCHEMA.INNODB\_METRICS**

**WHERE NAME like '%index\_page\_merge%';**

+-----------------------------+----------------------------------------+

| NAME | COMMENT |

+-----------------------------+----------------------------------------+

| index\_page\_merge\_attempts | Number of index page merge attempts |

| index\_page\_merge\_successful | Number of successful index page merges |

+-----------------------------+----------------------------------------+

When lowering the **MERGE\_THRESHOLD** value, the objectives are:

A smaller number of page merge attempts and successful page merges

A similar number of page merge attempts and successful page merges

A **MERGE\_THRESHOLD** setting that is too small could result in large data files due to an excessive amount of empty page space.

For information about using [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) counters, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

### 15.8.12 Enabling Automatic Configuration for a Dedicated MySQL Server

When [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled, **InnoDB** automatically configures the following variables:

[**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size)

[**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size)

[**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) (as of MySQL 8.0.14)

[**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method)

Only consider enabling [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) if the MySQL instance resides on a dedicated server where it can use all available system resources. For example, consider enabling if you run MySQL Server in a Docker container or dedicated VM that only runs MySQL. Enabling [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is not recommended if the MySQL instance shares system resources with other applications.

The information that follows describes how each variable is automatically configured.

[**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size)

Buffer pool size is configured according to the amount of memory detected on the server.

**Table 15.8 Automatically Configured Buffer Pool Size**

| **Detected Server Memory** | **Buffer Pool Size** |
| --- | --- |
| Less than 1GB | 128MiB (the default value) |
| 1GB to 4GB | ***detected server memory*** \* 0.5 |
| Greater than 4GB | ***detected server memory*** \* 0.75 |

[**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size)

As of MySQL 8.0.14, log file size is configured according to the automatically configured buffer pool size.

**Table 15.9 Automatically Configured Log File Size**

| **Buffer Pool Size** | **Log File Size** |
| --- | --- |
| Less than 8GB | 512MiB |
| 8GB to 128GB | 1024MiB |
| Greater than 128GB | 2048MiB |

**Note**

Prior to MySQL 8.0.14, the [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) variable was automatically configured according to the amount of memory detected on the server, as shown below:

**Table 15.10 Automatically Configured Log File Size (MySQL 8.0.13 and Earlier)**

| **Detected Server Memory** | **Log File Size** |
| --- | --- |
| < 1GB | 48MiB (the default value) |
| <= 4GB | 128MiB |
| <= 8GB | 512MiB |
| <= 16GB | 1024MiB |
| > 16GB | 2048MiB |

[**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group)

The number of log files is configured according to the automatically configured buffer pool size (in gigabytes). Automatic configuration of the [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) variable was added in MySQL 8.0.14.

**Table 15.11 Automatically Configured Number of Log Files**

| **Buffer Pool Size** | **Number of Log Files** |
| --- | --- |
| Less than 8GB | ROUND(***buffer pool size***) |
| 8GB to 128GB | ROUND(***buffer pool size*** \* 0.75) |
| Greater than 128GB | 64 |

**Note**

The minimum [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) value of 2 is enforced if the rounded buffer pool size value is less than 2GB.

[**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method)

The flush method is set to **O\_DIRECT\_NO\_FSYNC** when [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled. If the **O\_DIRECT\_NO\_FSYNC** setting is not available, the default [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) setting is used.

**InnoDB** uses **O\_DIRECT** during flushing I/O, but skips the **fsync()** system call after each write operation.

**Warning**

Prior to MySQL 8.0.14, this setting is not suitable for file systems such as XFS and EXT4, which require an **fsync()** system call to synchronize file system metadata changes.

As of MySQL 8.0.14, **fsync()** is called after creating a new file, after increasing file size, and after closing a file, to ensure that file system metadata changes are synchronized. The **fsync()** system call is still skipped after each write operation.

Data loss is possible if redo log files and data files reside on different storage devices, and an unexpected exit occurs before data file writes are flushed from a device cache that is not battery-backed. If you use or intend to use different storage devices for redo log files and data files, and your data files reside on a device with a cache that is not battery-backed, use **O\_DIRECT** instead.

If an automatically configured option is configured explicitly in an option file or elsewhere, the explicitly specified setting is used, and a startup warning similar to this is printed to **stderr**:

[Warning] [000000] InnoDB: Option innodb\_dedicated\_server is ignored for innodb\_buffer\_pool\_size because innodb\_buffer\_pool\_size=134217728 is specified explicitly.

Explicit configuration of one option does not prevent the automatic configuration of other options.

If [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled and [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is configured explicitly in an option file, [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) and [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) are still automatically configured based on a buffer pool size value calculated according to the amount of memory detected on the server, even though that value is not used to configure the size of the buffer pool.

Automatically configured settings are evaluated and reconfigured if necessary each time the MySQL server is started.

## 15.9 InnoDB Table and Page Compression

[15.9.1 InnoDB Table Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-compression)

[15.9.2 InnoDB Page Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-page-compression)

This section provides information about the **InnoDB** table compression and **InnoDB** page compression features. The page compression feature is also referred to as [transparent page compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transparent_page_compression).

Using the compression features of **InnoDB**, you can create tables where the data is stored in compressed form. Compression can help to improve both raw performance and scalability. The compression means less data is transferred between disk and memory, and takes up less space on disk and in memory. The benefits are amplified for tables with [secondary indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index), because index data is compressed also. Compression can be especially important for [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) storage devices, because they tend to have lower capacity than [HDD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_hdd) devices.

### 15.9.1 InnoDB Table Compression

[15.9.1.1 Overview of Table Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-background)

[15.9.1.2 Creating Compressed Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-usage)

[15.9.1.3 Tuning Compression for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning)

[15.9.1.4 Monitoring InnoDB Table Compression at Runtime](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning-monitoring)

[15.9.1.5 How Compression Works for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals)

[15.9.1.6 Compression for OLTP Workloads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp)

[15.9.1.7 SQL Compression Syntax Warnings and Errors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-syntax-warnings)

This section describes **InnoDB** table compression, which is supported with **InnoDB** tables that reside in [file\_per\_table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces or [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace). Table compression is enabled using the **ROW\_FORMAT=COMPRESSED** attribute with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

#### 15.9.1.1 Overview of Table Compression

Because processors and cache memories have increased in speed more than disk storage devices, many workloads are [disk-bound](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_disk_bound). Data [compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) enables smaller database size, reduced I/O, and improved throughput, at the small cost of increased CPU utilization. Compression is especially valuable for read-intensive applications, on systems with enough RAM to keep frequently used data in memory.

An **InnoDB** table created with **ROW\_FORMAT=COMPRESSED** can use a smaller [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size) on disk than the configured [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. Smaller pages require less I/O to read from and write to disk, which is especially valuable for [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) devices.

The compressed page size is specified through the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) **KEY\_BLOCK\_SIZE** parameter. The different page size requires that the table be placed in a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace or [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) rather than in the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace), as the system tablespace cannot store compressed tables. For more information, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces), and [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

The level of compression is the same regardless of the **KEY\_BLOCK\_SIZE** value. As you specify smaller values for **KEY\_BLOCK\_SIZE**, you get the I/O benefits of increasingly smaller pages. But if you specify a value that is too small, there is additional overhead to reorganize the pages when data values cannot be compressed enough to fit multiple rows in each page. There is a hard limit on how small **KEY\_BLOCK\_SIZE** can be for a table, based on the lengths of the key columns for each of its indexes. Specify a value that is too small, and the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement fails.

In the buffer pool, the compressed data is held in small pages, with a page size based on the **KEY\_BLOCK\_SIZE** value. For extracting or updating the column values, MySQL also creates an uncompressed page in the buffer pool with the uncompressed data. Within the buffer pool, any updates to the uncompressed page are also re-written back to the equivalent compressed page. You might need to size your buffer pool to accommodate the additional data of both compressed and uncompressed pages, although the uncompressed pages are [evicted](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction) from the buffer pool when space is needed, and then uncompressed again on the next access.

#### 15.9.1.2 Creating Compressed Tables

Compressed tables can be created in [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces or in [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace). Table compression is not available for the InnoDB [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace). The system tablespace (space 0, the [.ibdata files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file)) can contain user-created tables, but it also contains internal system data, which is never compressed. Thus, compression applies only to tables (and indexes) stored in file-per-table or general tablespaces.

##### Creating a Compressed Table in File-Per-Table Tablespace

To create a compressed table in a file-per-table tablespace, [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) must be enabled (the default). You can set this parameter in the MySQL configuration file (my.cnf or my.ini) or dynamically, using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement.

After the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) option is configured, specify the **ROW\_FORMAT=COMPRESSED** clause or **KEY\_BLOCK\_SIZE** clause, or both, in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to create a compressed table in a file-per-table tablespace.

For example, you might use the following statements:

SET GLOBAL innodb\_file\_per\_table=1;

CREATE TABLE t1

(c1 INT PRIMARY KEY)

ROW\_FORMAT=COMPRESSED

KEY\_BLOCK\_SIZE=8;

##### Creating a Compressed Table in a General Tablespace

To create a compressed table in a general tablespace, **FILE\_BLOCK\_SIZE** must be defined for the general tablespace, which is specified when the tablespace is created. The **FILE\_BLOCK\_SIZE** value must be a valid compressed page size in relation to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value, and the page size of the compressed table, defined by the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) **KEY\_BLOCK\_SIZE** clause, must be equal to **FILE\_BLOCK\_SIZE/1024**. For example, if [**innodb\_page\_size=16384**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) and **FILE\_BLOCK\_SIZE=8192**, the **KEY\_BLOCK\_SIZE** of the table must be 8. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

The following example demonstrates creating a general tablespace and adding a compressed table. The example assumes a default [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) of 16K. The **FILE\_BLOCK\_SIZE** of 8192 requires that the compressed table have a **KEY\_BLOCK\_SIZE** of 8.

mysql> **CREATE TABLESPACE `ts2` ADD DATAFILE 'ts2.ibd' FILE\_BLOCK\_SIZE = 8192 Engine=InnoDB;**

mysql> **CREATE TABLE t4 (c1 INT PRIMARY KEY) TABLESPACE ts2 ROW\_FORMAT=COMPRESSED KEY\_BLOCK\_SIZE=8;**

##### Notes

As of MySQL 8.0, the tablespace file for a compressed table is created using the physical page size instead of the **InnoDB** page size, which makes the initial size of a tablespace file for an empty compressed table smaller than in previous MySQL releases.

If you specify **ROW\_FORMAT=COMPRESSED**, you can omit **KEY\_BLOCK\_SIZE**; the **KEY\_BLOCK\_SIZE** setting defaults to half the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value.

If you specify a valid **KEY\_BLOCK\_SIZE** value, you can omit **ROW\_FORMAT=COMPRESSED**; compression is enabled automatically.

To determine the best value for **KEY\_BLOCK\_SIZE,** typically you create several copies of the same table with different values for this clause, then measure the size of the resulting .ibd files and see how well each performs with a realistic [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload). For general tablespaces, keep in mind that dropping a table does not reduce the size of the general tablespace .ibd file, nor does it return disk space to the operating system. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

The **KEY\_BLOCK\_SIZE** value is treated as a hint; a different size could be used by **InnoDB** if necessary. For file-per-table tablespaces, the **KEY\_BLOCK\_SIZE** can only be less than or equal to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. If you specify a value greater than the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value, the specified value is ignored, a warning is issued, and **KEY\_BLOCK\_SIZE** is set to half of the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. If **innodb\_strict\_mode=ON**, specifying an invalid **KEY\_BLOCK\_SIZE** value returns an error. For general tablespaces, valid **KEY\_BLOCK\_SIZE** values depend on the **FILE\_BLOCK\_SIZE** setting of the tablespace. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

**InnoDB** supports 32KB and 64KB page sizes but these page sizes do not support compression. For more information, refer to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) documentation.

The default uncompressed size of **InnoDB** data [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) is 16KB. Depending on the combination of option values, MySQL uses a page size of 1KB, 2KB, 4KB, 8KB, or 16KB for the tablespace data file (.ibd file). The actual compression algorithm is not affected by the **KEY\_BLOCK\_SIZE** value; the value determines how large each compressed chunk is, which in turn affects how many rows can be packed into each compressed page.

When creating a compressed table in a file-per-table tablespace, setting **KEY\_BLOCK\_SIZE** equal to the **InnoDB** [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size) does not typically result in much compression. For example, setting **KEY\_BLOCK\_SIZE=16** typically would not result in much compression, since the normal **InnoDB** page size is 16KB. This setting may still be useful for tables with many long [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, because such values often do compress well, and might therefore require fewer [overflow pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_overflow_page) as described in [Section 15.9.1.5, “How Compression Works for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals). For general tablespaces, a **KEY\_BLOCK\_SIZE** value equal to the **InnoDB** page size is not permitted. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

All indexes of a table (including the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index)) are compressed using the same page size, as specified in the **CREATE TABLE** or **ALTER TABLE** statement. Table attributes such as **ROW\_FORMAT** and **KEY\_BLOCK\_SIZE** are not part of the **CREATE INDEX** syntax for **InnoDB** tables, and are ignored if they are specified (although, if specified, they appear in the output of the [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) statement).

For performance-related configuration options, see [Section 15.9.1.3, “Tuning Compression for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning).

##### Restrictions on Compressed Tables

Compressed tables cannot be stored in the **InnoDB** system tablespace.

General tablespaces can contain multiple tables, but compressed and uncompressed tables cannot coexist within the same general tablespace.

Compression applies to an entire table and all its associated indexes, not to individual rows, despite the clause name **ROW\_FORMAT**.

**InnoDB** does not support compressed temporary tables. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled (the default), [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) returns errors if **ROW\_FORMAT=COMPRESSED** or **KEY\_BLOCK\_SIZE** is specified. If [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, warnings are issued and the temporary table is created using a non-compressed row format. The same restrictions apply to [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations on temporary tables.

#### 15.9.1.3 Tuning Compression for InnoDB Tables

Most often, the internal optimizations described in [InnoDB Data Storage and Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals-storage) ensure that the system runs well with compressed data. However, because the efficiency of compression depends on the nature of your data, you can make decisions that affect the performance of compressed tables:

Which tables to compress.

What compressed page size to use.

Whether to adjust the size of the buffer pool based on run-time performance characteristics, such as the amount of time the system spends compressing and uncompressing data. Whether the workload is more like a [data warehouse](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_warehouse) (primarily queries) or an [OLTP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_oltp) system (mix of queries and [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml)).

If the system performs DML operations on compressed tables, and the way the data is distributed leads to expensive [compression failures](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure) at runtime, you might adjust additional advanced configuration options.

Use the guidelines in this section to help make those architectural and configuration choices. When you are ready to conduct long-term testing and put compressed tables into production, see [Section 15.9.1.4, “Monitoring InnoDB Table Compression at Runtime”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning-monitoring) for ways to verify the effectiveness of those choices under real-world conditions.

##### When to Use Compression

In general, compression works best on tables that include a reasonable number of character string columns and where the data is read far more often than it is written. Because there are no guaranteed ways to predict whether or not compression benefits a particular situation, always test with a specific [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload) and data set running on a representative configuration. Consider the following factors when deciding which tables to compress.

##### Data Characteristics and Compression

A key determinant of the efficiency of compression in reducing the size of data files is the nature of the data itself. Recall that compression works by identifying repeated strings of bytes in a block of data. Completely randomized data is the worst case. Typical data often has repeated values, and so compresses effectively. Character strings often compress well, whether defined in **CHAR**, **VARCHAR**, **TEXT** or **BLOB** columns. On the other hand, tables containing mostly binary data (integers or floating point numbers) or data that is previously compressed (for example JPEG or PNG images) may not generally compress well, significantly or at all.

You choose whether to turn on compression for each InnoDB table. A table and all of its indexes use the same (compressed) [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size). It might be that the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) (clustered) index, which contains the data for all columns of a table, compresses more effectively than the secondary indexes. For those cases where there are long rows, the use of compression might result in long column values being stored “off-page”, as discussed in [DYNAMIC Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-dynamic). Those overflow pages may compress well. Given these considerations, for many applications, some tables compress more effectively than others, and you might find that your workload performs best only with a subset of tables compressed.

To determine whether or not to compress a particular table, conduct experiments. You can get a rough estimate of how efficiently your data can be compressed by using a utility that implements LZ77 compression (such as **gzip** or WinZip) on a copy of the [.ibd file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) for an uncompressed table. You can expect less compression from a MySQL compressed table than from file-based compression tools, because MySQL compresses data in chunks based on the [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size), 16KB by default. In addition to user data, the page format includes some internal system data that is not compressed. File-based compression utilities can examine much larger chunks of data, and so might find more repeated strings in a huge file than MySQL can find in an individual page.

Another way to test compression on a specific table is to copy some data from your uncompressed table to a similar, compressed table (having all the same indexes) in a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace and look at the size of the resulting **.ibd** file. For example:

USE test;

SET GLOBAL innodb\_file\_per\_table=1;

SET GLOBAL autocommit=0;

-- Create an uncompressed table with a million or two rows.

CREATE TABLE big\_table AS SELECT \* FROM information\_schema.columns;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

INSERT INTO big\_table SELECT \* FROM big\_table;

COMMIT;

ALTER TABLE big\_table ADD id int unsigned NOT NULL PRIMARY KEY auto\_increment;

SHOW CREATE TABLE big\_table\G

select count(id) from big\_table;

-- Check how much space is needed for the uncompressed table.

\! ls -l data/test/big\_table.ibd

CREATE TABLE key\_block\_size\_4 LIKE big\_table;

ALTER TABLE key\_block\_size\_4 key\_block\_size=4 row\_format=compressed;

INSERT INTO key\_block\_size\_4 SELECT \* FROM big\_table;

commit;

-- Check how much space is needed for a compressed table

-- with particular compression settings.

\! ls -l data/test/key\_block\_size\_4.ibd

This experiment produced the following numbers, which of course could vary considerably depending on your table structure and data:

-rw-rw---- 1 cirrus staff 310378496 Jan 9 13:44 data/test/big\_table.ibd

-rw-rw---- 1 cirrus staff 83886080 Jan 9 15:10 data/test/key\_block\_size\_4.ibd

To see whether compression is efficient for your particular [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload):

For simple tests, use a MySQL instance with no other compressed tables and run queries against the [**INFORMATION\_SCHEMA.INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) table.

For more elaborate tests involving workloads with multiple compressed tables, run queries against the [**INFORMATION\_SCHEMA.INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table. Because the statistics in the **INNODB\_CMP\_PER\_INDEX** table are expensive to collect, you must enable the configuration option [**innodb\_cmp\_per\_index\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_cmp_per_index_enabled) before querying that table, and you might restrict such testing to a development server or a non-critical replica server.

Run some typical SQL statements against the compressed table you are testing.

Examine the ratio of successful compression operations to overall compression operations by querying the [**INFORMATION\_SCHEMA.INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) or [**INFORMATION\_SCHEMA.INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table, and comparing **COMPRESS\_OPS** to **COMPRESS\_OPS\_OK**.

If a high percentage of compression operations complete successfully, the table might be a good candidate for compression.

If you get a high proportion of [compression failures](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure), you can adjust [**innodb\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_level), [**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct), and [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max) options as described in [Section 15.9.1.6, “Compression for OLTP Workloads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp), and try further tests.

##### Database Compression versus Application Compression

Decide whether to compress data in your application or in the table; do not use both types of compression for the same data. When you compress the data in the application and store the results in a compressed table, extra space savings are extremely unlikely, and the double compression just wastes CPU cycles.

##### Compressing in the Database

When enabled, MySQL table compression is automatic and applies to all columns and index values. The columns can still be tested with operators such as **LIKE**, and sort operations can still use indexes even when the index values are compressed. Because indexes are often a significant fraction of the total size of a database, compression could result in significant savings in storage, I/O or processor time. The compression and decompression operations happen on the database server, which likely is a powerful system that is sized to handle the expected load.

##### Compressing in the Application

If you compress data such as text in your application, before it is inserted into the database, You might save overhead for data that does not compress well by compressing some columns and not others. This approach uses CPU cycles for compression and uncompression on the client machine rather than the database server, which might be appropriate for a distributed application with many clients, or where the client machine has spare CPU cycles.

##### Hybrid Approach

Of course, it is possible to combine these approaches. For some applications, it may be appropriate to use some compressed tables and some uncompressed tables. It may be best to externally compress some data (and store it in uncompressed tables) and allow MySQL to compress (some of) the other tables in the application. As always, up-front design and real-life testing are valuable in reaching the right decision.

##### Workload Characteristics and Compression

In addition to choosing which tables to compress (and the page size), the workload is another key determinant of performance. If the application is dominated by reads, rather than updates, fewer pages need to be reorganized and recompressed after the index page runs out of room for the per-page “modification log” that MySQL maintains for compressed data. If the updates predominantly change non-indexed columns or those containing **BLOB**s or large strings that happen to be stored “off-page”, the overhead of compression may be acceptable. If the only changes to a table are **INSERT**s that use a monotonically increasing primary key, and there are few secondary indexes, there is little need to reorganize and recompress index pages. Since MySQL can “delete-mark” and delete rows on compressed pages “in place” by modifying uncompressed data, **DELETE** operations on a table are relatively efficient.

For some environments, the time it takes to load data can be as important as run-time retrieval. Especially in data warehouse environments, many tables may be read-only or read-mostly. In those cases, it might or might not be acceptable to pay the price of compression in terms of increased load time, unless the resulting savings in fewer disk reads or in storage cost is significant.

Fundamentally, compression works best when the CPU time is available for compressing and uncompressing data. Thus, if your workload is I/O bound, rather than CPU-bound, you might find that compression can improve overall performance. When you test your application performance with different compression configurations, test on a platform similar to the planned configuration of the production system.

##### Configuration Characteristics and Compression

Reading and writing database [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) from and to disk is the slowest aspect of system performance. Compression attempts to reduce I/O by using CPU time to compress and uncompress data, and is most effective when I/O is a relatively scarce resource compared to processor cycles.

This is often especially the case when running in a multi-user environment with fast, multi-core CPUs. When a page of a compressed table is in memory, MySQL often uses additional memory, typically 16KB, in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) for an uncompressed copy of the page. The adaptive LRU algorithm attempts to balance the use of memory between compressed and uncompressed pages to take into account whether the workload is running in an I/O-bound or CPU-bound manner. Still, a configuration with more memory dedicated to the buffer pool tends to run better when using compressed tables than a configuration where memory is highly constrained.

##### Choosing the Compressed Page Size

The optimal setting of the compressed page size depends on the type and distribution of data that the table and its indexes contain. The compressed page size should always be bigger than the maximum record size, or operations may fail as noted in [Compression of B-Tree Pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals-storage-btree).

Setting the compressed page size too large wastes some space, but the pages do not have to be compressed as often. If the compressed page size is set too small, inserts or updates may require time-consuming recompression, and the [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) nodes may have to be split more frequently, leading to bigger data files and less efficient indexing.

Typically, you set the compressed page size to 8K or 4K bytes. Given that the maximum row size for an InnoDB table is around 8K, **KEY\_BLOCK\_SIZE=8** is usually a safe choice.

#### 15.9.1.4 Monitoring InnoDB Table Compression at Runtime

Overall application performance, CPU and I/O utilization and the size of disk files are good indicators of how effective compression is for your application. This section builds on the performance tuning advice from [Section 15.9.1.3, “Tuning Compression for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning), and shows how to find problems that might not turn up during initial testing.

To dig deeper into performance considerations for compressed tables, you can monitor compression performance at runtime using the [Information Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_information_schema) tables described in [Example 15.1, “Using the Compression Information Schema Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-examples-compression). These tables reflect the internal use of memory and the rates of compression used overall.

The [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) table reports information about compression activity for each compressed page size (**KEY\_BLOCK\_SIZE**) in use. The information in these tables is system-wide: it summarizes the compression statistics across all compressed tables in your database. You can use this data to help decide whether or not to compress a table by examining these tables when no other compressed tables are being accessed. It involves relatively low overhead on the server, so you might query it periodically on a production server to check the overall efficiency of the compression feature.

The [**INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table reports information about compression activity for individual tables and indexes. This information is more targeted and more useful for evaluating compression efficiency and diagnosing performance issues one table or index at a time. (Because that each **InnoDB** table is represented as a clustered index, MySQL does not make a big distinction between tables and indexes in this context.) The [**INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table does involve substantial overhead, so it is more suitable for development servers, where you can compare the effects of different [workloads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload), data, and compression settings in isolation. To guard against imposing this monitoring overhead by accident, you must enable the [**innodb\_cmp\_per\_index\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_cmp_per_index_enabled) configuration option before you can query the [**INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table.

The key statistics to consider are the number of, and amount of time spent performing, compression and uncompression operations. Since MySQL splits [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) nodes when they are too full to contain the compressed data following a modification, compare the number of “successful” compression operations with the number of such operations overall. Based on the information in the [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) and [**INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) tables and overall application performance and hardware resource utilization, you might make changes in your hardware configuration, adjust the size of the buffer pool, choose a different page size, or select a different set of tables to compress.

If the amount of CPU time required for compressing and uncompressing is high, changing to faster or multi-core CPUs can help improve performance with the same data, application workload and set of compressed tables. Increasing the size of the buffer pool might also help performance, so that more uncompressed pages can stay in memory, reducing the need to uncompress pages that exist in memory only in compressed form.

A large number of compression operations overall (compared to the number of **INSERT**, **UPDATE** and **DELETE** operations in your application and the size of the database) could indicate that some of your compressed tables are being updated too heavily for effective compression. If so, choose a larger page size, or be more selective about which tables you compress.

If the number of “successful” compression operations (**COMPRESS\_OPS\_OK**) is a high percentage of the total number of compression operations (**COMPRESS\_OPS**), then the system is likely performing well. If the ratio is low, then MySQL is reorganizing, recompressing, and splitting B-tree nodes more often than is desirable. In this case, avoid compressing some tables, or increase **KEY\_BLOCK\_SIZE** for some of the compressed tables. You might turn off compression for tables that cause the number of “compression failures” in your application to be more than 1% or 2% of the total. (Such a failure ratio might be acceptable during a temporary operation such as a data load).

#### 15.9.1.5 How Compression Works for InnoDB Tables

This section describes some internal implementation details about [compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) for InnoDB tables. The information presented here may be helpful in tuning for performance, but is not necessary to know for basic use of compression.

##### Compression Algorithms

Some operating systems implement compression at the file system level. Files are typically divided into fixed-size blocks that are compressed into variable-size blocks, which easily leads into fragmentation. Every time something inside a block is modified, the whole block is recompressed before it is written to disk. These properties make this compression technique unsuitable for use in an update-intensive database system.

MySQL implements compression with the help of the well-known [zlib library](http://www.zlib.net/), which implements the LZ77 compression algorithm. This compression algorithm is mature, robust, and efficient in both CPU utilization and in reduction of data size. The algorithm is “lossless”, so that the original uncompressed data can always be reconstructed from the compressed form. LZ77 compression works by finding sequences of data that are repeated within the data to be compressed. The patterns of values in your data determine how well it compresses, but typical user data often compresses by 50% or more.

**Note**

**InnoDB** supports the **zlib** library up to version 1.2.11, which is the version bundled with MySQL 8.0.

Unlike compression performed by an application, or compression features of some other database management systems, InnoDB compression applies both to user data and to indexes. In many cases, indexes can constitute 40-50% or more of the total database size, so this difference is significant. When compression is working well for a data set, the size of the InnoDB data files (the [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace or [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) **.ibd** files) is 25% to 50% of the uncompressed size or possibly smaller. Depending on the [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload), this smaller database can in turn lead to a reduction in I/O, and an increase in throughput, at a modest cost in terms of increased CPU utilization. You can adjust the balance between compression level and CPU overhead by modifying the [**innodb\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_level) configuration option.

##### InnoDB Data Storage and Compression

All user data in InnoDB tables is stored in pages comprising a [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) index (the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index)). In some other database systems, this type of index is called an “index-organized table”. Each row in the index node contains the values of the (user-specified or system-generated) [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) and all the other columns of the table.

[Secondary indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index) in InnoDB tables are also B-trees, containing pairs of values: the index key and a pointer to a row in the clustered index. The pointer is in fact the value of the primary key of the table, which is used to access the clustered index if columns other than the index key and primary key are required. Secondary index records must always fit on a single B-tree page.

The compression of B-tree nodes (of both clustered and secondary indexes) is handled differently from compression of [overflow pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_overflow_page) used to store long **VARCHAR**, **BLOB**, or **TEXT** columns, as explained in the following sections.

##### Compression of B-Tree Pages

Because they are frequently updated, B-tree pages require special treatment. It is important to minimize the number of times B-tree nodes are split, as well as to minimize the need to uncompress and recompress their content.

One technique MySQL uses is to maintain some system information in the B-tree node in uncompressed form, thus facilitating certain in-place updates. For example, this allows rows to be delete-marked and deleted without any compression operation.

In addition, MySQL attempts to avoid unnecessary uncompression and recompression of index pages when they are changed. Within each B-tree page, the system keeps an uncompressed “modification log” to record changes made to the page. Updates and inserts of small records may be written to this modification log without requiring the entire page to be completely reconstructed.

When the space for the modification log runs out, InnoDB uncompresses the page, applies the changes and recompresses the page. If recompression fails (a situation known as a [compression failure](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure)), the B-tree nodes are split and the process is repeated until the update or insert succeeds.

To avoid frequent compression failures in write-intensive workloads, such as for [OLTP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_oltp) applications, MySQL sometimes reserves some empty space (padding) in the page, so that the modification log fills up sooner and the page is recompressed while there is still enough room to avoid splitting it. The amount of padding space left in each page varies as the system keeps track of the frequency of page splits. On a busy server doing frequent writes to compressed tables, you can adjust the [**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct), and [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max) configuration options to fine-tune this mechanism.

Generally, MySQL requires that each B-tree page in an InnoDB table can accommodate at least two records. For compressed tables, this requirement has been relaxed. Leaf pages of B-tree nodes (whether of the primary key or secondary indexes) only need to accommodate one record, but that record must fit, in uncompressed form, in the per-page modification log. If [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is **ON**, MySQL checks the maximum row size during [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index). If the row does not fit, the following error message is issued: **ERROR HY000: Too big row**.

If you create a table when [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is OFF, and a subsequent **INSERT** or **UPDATE** statement attempts to create an index entry that does not fit in the size of the compressed page, the operation fails with **ERROR 42000: Row size too large**. (This error message does not name the index for which the record is too large, or mention the length of the index record or the maximum record size on that particular index page.) To solve this problem, rebuild the table with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and select a larger compressed page size (**KEY\_BLOCK\_SIZE**), shorten any column prefix indexes, or disable compression entirely with **ROW\_FORMAT=DYNAMIC** or **ROW\_FORMAT=COMPACT**.

[**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is not applicable to general tablespaces, which also support compressed tables. Tablespace management rules for general tablespaces are strictly enforced independently of [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode). For more information, see [Section 13.1.21, “CREATE TABLESPACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace).

##### Compressing BLOB, VARCHAR, and TEXT Columns

In an InnoDB table, [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns that are not part of the primary key may be stored on separately allocated [overflow pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_overflow_page). We refer to these columns as [off-page columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_off_page_column). Their values are stored on singly-linked lists of overflow pages.

For tables created in **ROW\_FORMAT=DYNAMIC** or **ROW\_FORMAT=COMPRESSED**, the values of [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns may be stored fully off-page, depending on their length and the length of the entire row. For columns that are stored off-page, the clustered index record only contains 20-byte pointers to the overflow pages, one per column. Whether any columns are stored off-page depends on the page size and the total size of the row. When the row is too long to fit entirely within the page of the clustered index, MySQL chooses the longest columns for off-page storage until the row fits on the clustered index page. As noted above, if a row does not fit by itself on a compressed page, an error occurs.

**Note**

For tables created in **ROW\_FORMAT=DYNAMIC** or **ROW\_FORMAT=COMPRESSED**, [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns that are less than or equal to 40 bytes are always stored in-line.

Tables that use **ROW\_FORMAT=REDUNDANT** and **ROW\_FORMAT=COMPACT** store the first 768 bytes of [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns in the clustered index record along with the primary key. The 768-byte prefix is followed by a 20-byte pointer to the overflow pages that contain the rest of the column value.

When a table is in **COMPRESSED** format, all data written to overflow pages is compressed “as is”; that is, MySQL applies the zlib compression algorithm to the entire data item. Other than the data, compressed overflow pages contain an uncompressed header and trailer comprising a page checksum and a link to the next overflow page, among other things. Therefore, very significant storage savings can be obtained for longer **BLOB**, **TEXT**, or **VARCHAR** columns if the data is highly compressible, as is often the case with text data. Image data, such as **JPEG**, is typically already compressed and so does not benefit much from being stored in a compressed table; the double compression can waste CPU cycles for little or no space savings.

The overflow pages are of the same size as other pages. A row containing ten columns stored off-page occupies ten overflow pages, even if the total length of the columns is only 8K bytes. In an uncompressed table, ten uncompressed overflow pages occupy 160K bytes. In a compressed table with an 8K page size, they occupy only 80K bytes. Thus, it is often more efficient to use compressed table format for tables with long column values.

For [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces, using a 16K compressed page size can reduce storage and I/O costs for [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, because such data often compress well, and might therefore require fewer overflow pages, even though the B-tree nodes themselves take as many pages as in the uncompressed form. General tablespaces do not support a 16K compressed page size (**KEY\_BLOCK\_SIZE**). For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

##### Compression and the InnoDB Buffer Pool

In a compressed **InnoDB** table, every compressed page (whether 1K, 2K, 4K or 8K) corresponds to an uncompressed page of 16K bytes (or a smaller size if [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is set). To access the data in a page, MySQL reads the compressed page from disk if it is not already in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), then uncompresses the page to its original form. This section describes how **InnoDB** manages the buffer pool with respect to pages of compressed tables.

To minimize I/O and to reduce the need to uncompress a page, at times the buffer pool contains both the compressed and uncompressed form of a database page. To make room for other required database pages, MySQL can [evict](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction) from the buffer pool an uncompressed page, while leaving the compressed page in memory. Or, if a page has not been accessed in a while, the compressed form of the page might be written to disk, to free space for other data. Thus, at any given time, the buffer pool might contain both the compressed and uncompressed forms of the page, or only the compressed form of the page, or neither.

MySQL keeps track of which pages to keep in memory and which to evict using a least-recently-used ([LRU](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lru)) list, so that [hot](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_hot) (frequently accessed) data tends to stay in memory. When compressed tables are accessed, MySQL uses an adaptive LRU algorithm to achieve an appropriate balance of compressed and uncompressed pages in memory. This adaptive algorithm is sensitive to whether the system is running in an [I/O-bound](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_io_bound) or [CPU-bound](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cpu_bound) manner. The goal is to avoid spending too much processing time uncompressing pages when the CPU is busy, and to avoid doing excess I/O when the CPU has spare cycles that can be used for uncompressing compressed pages (that may already be in memory). When the system is I/O-bound, the algorithm prefers to evict the uncompressed copy of a page rather than both copies, to make more room for other disk pages to become memory resident. When the system is CPU-bound, MySQL prefers to evict both the compressed and uncompressed page, so that more memory can be used for “hot” pages and reducing the need to uncompress data in memory only in compressed form.

##### Compression and the InnoDB Redo Log Files

Before a compressed page is written to a [data file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_files), MySQL writes a copy of the page to the redo log (if it has been recompressed since the last time it was written to the database). This is done to ensure that redo logs are usable for [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery), even in the unlikely case that the **zlib** library is upgraded and that change introduces a compatibility problem with the compressed data. Therefore, some increase in the size of [log files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file), or a need for more frequent [checkpoints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint), can be expected when using compression. The amount of increase in the log file size or checkpoint frequency depends on the number of times compressed pages are modified in a way that requires reorganization and recompression.

To create a compressed table in a file-per-table tablespace, [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) must be enabled. There is no dependence on the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting when creating a compressed table in a general tablespace. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

#### 15.9.1.6 Compression for OLTP Workloads

Traditionally, the **InnoDB** [compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) feature was recommended primarily for read-only or read-mostly [workloads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload), such as in a [data warehouse](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_warehouse) configuration. The rise of [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) storage devices, which are fast but relatively small and expensive, makes compression attractive also for **OLTP** workloads: high-traffic, interactive websites can reduce their storage requirements and their I/O operations per second ([IOPS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_iops)) by using compressed tables with applications that do frequent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations.

These configuration options let you adjust the way compression works for a particular MySQL instance, with an emphasis on performance and scalability for write-intensive operations:

[**innodb\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_level) lets you turn the degree of compression up or down. A higher value lets you fit more data onto a storage device, at the expense of more CPU overhead during compression. A lower value lets you reduce CPU overhead when storage space is not critical, or you expect the data is not especially compressible.

[**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct) specifies a cutoff point for [compression failures](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure) during updates to a compressed table. When this threshold is passed, MySQL begins to leave additional free space within each new compressed page, dynamically adjusting the amount of free space up to the percentage of page size specified by [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max)

[**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max) lets you adjust the maximum amount of space reserved within each [page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) to record changes to compressed rows, without needing to compress the entire page again. The higher the value, the more changes can be recorded without recompressing the page. MySQL uses a variable amount of free space for the pages within each compressed table, only when a designated percentage of compression operations “[fail](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure)” at runtime, requiring an expensive operation to split the compressed page.

[**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) lets you disable writing of images of [re-compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) to the [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log). Re-compression may occur when changes are made to compressed data. This option is enabled by default to prevent corruption that could occur if a different version of the **zlib** compression algorithm is used during recovery. If you are certain that the **zlib** version is not subject to change, disable [**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) to reduce redo log generation for workloads that modify compressed data.

Because working with compressed data sometimes involves keeping both compressed and uncompressed versions of a page in memory at the same time, when using compression with an OLTP-style workload, be prepared to increase the value of the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) configuration option.

#### 15.9.1.7 SQL Compression Syntax Warnings and Errors

This section describes syntax warnings and errors that you may encounter when using the table compression feature with [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces and [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace).

##### SQL Compression Syntax Warnings and Errors for File-Per-Table Tablespaces

When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled (the default), specifying **ROW\_FORMAT=COMPRESSED** or **KEY\_BLOCK\_SIZE** in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements produces the following error if [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is disabled.

ERROR 1031 (HY000): Table storage engine for 't1' doesn't have this option

**Note**

The table is not created if the current configuration does not permit using compressed tables.

When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, specifying **ROW\_FORMAT=COMPRESSED** or **KEY\_BLOCK\_SIZE** in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements produces the following warnings if [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is disabled.

mysql> **SHOW WARNINGS;**

+---------+------+---------------------------------------------------------------+

| Level | Code | Message |

+---------+------+---------------------------------------------------------------+

| Warning | 1478 | InnoDB: KEY\_BLOCK\_SIZE requires innodb\_file\_per\_table. |

| Warning | 1478 | InnoDB: ignoring KEY\_BLOCK\_SIZE=4. |

| Warning | 1478 | InnoDB: ROW\_FORMAT=COMPRESSED requires innodb\_file\_per\_table. |

| Warning | 1478 | InnoDB: assuming ROW\_FORMAT=DYNAMIC. |

+---------+------+---------------------------------------------------------------+

**Note**

These messages are only warnings, not errors, and the table is created without compression, as if the options were not specified.

The “non-strict” behavior lets you import a **mysqldump** file into a database that does not support compressed tables, even if the source database contained compressed tables. In that case, MySQL creates the table in **ROW\_FORMAT=DYNAMIC** instead of preventing the operation.

To import the dump file into a new database, and have the tables re-created as they exist in the original database, ensure the server has the proper setting for the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) configuration parameter.

The attribute **KEY\_BLOCK\_SIZE** is permitted only when **ROW\_FORMAT** is specified as **COMPRESSED** or is omitted. Specifying a **KEY\_BLOCK\_SIZE** with any other **ROW\_FORMAT** generates a warning that you can view with **SHOW WARNINGS**. However, the table is non-compressed; the specified **KEY\_BLOCK\_SIZE** is ignored).

| **Level** | **Code** | **Message** |
| --- | --- | --- |
| **Warning** | 1478 | **InnoDB: ignoring KEY\_BLOCK\_SIZE=*n* unless ROW\_FORMAT=COMPRESSED.** |

If you are running with [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) enabled, the combination of a **KEY\_BLOCK\_SIZE** with any **ROW\_FORMAT** other than **COMPRESSED** generates an error, not a warning, and the table is not created.

[Table 15.12, “ROW\_FORMAT and KEY\_BLOCK\_SIZE Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-create-and-alter-options-table) provides an overview the **ROW\_FORMAT** and **KEY\_BLOCK\_SIZE** options that are used with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

**Table 15.12 ROW\_FORMAT and KEY\_BLOCK\_SIZE Options**

| **Option** | **Usage Notes** | **Description** |
| --- | --- | --- |
| **ROW\_FORMAT=​REDUNDANT** | Storage format used prior to MySQL 5.0.3 | Less efficient than **ROW\_FORMAT=COMPACT**; for backward compatibility |
| **ROW\_FORMAT=​COMPACT** | Default storage format since MySQL 5.0.3 | Stores a prefix of 768 bytes of long column values in the clustered index page, with the remaining bytes stored in an overflow page |
| **ROW\_FORMAT=​DYNAMIC** |  | Store values within the clustered index page if they fit; if not, stores only a 20-byte pointer to an overflow page (no prefix) |
| **ROW\_FORMAT=​COMPRESSED** |  | Compresses the table and indexes using zlib |
| **KEY\_BLOCK\_​SIZE=*n*** |  | Specifies compressed page size of 1, 2, 4, 8 or 16 kilobytes; implies **ROW\_FORMAT=COMPRESSED**. For general tablespaces, a **KEY\_BLOCK\_SIZE** value equal to the **InnoDB** page size is not permitted. |

[Table 15.13, “CREATE/ALTER TABLE Warnings and Errors when InnoDB Strict Mode is OFF”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-create-and-alter-errors-table) summarizes error conditions that occur with certain combinations of configuration parameters and options on the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements, and how the options appear in the output of **SHOW TABLE STATUS**.

When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is **OFF**, MySQL creates or alters the table, but ignores certain settings as shown below. You can see the warning messages in the MySQL error log. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is **ON**, these specified combinations of options generate errors, and the table is not created or altered. To see the full description of the error condition, issue the **SHOW ERRORS** statement: example:

mysql> **CREATE TABLE x (id INT PRIMARY KEY, c INT)**

-> **ENGINE=INNODB KEY\_BLOCK\_SIZE=33333;**

ERROR 1005 (HY000): Can't create table 'test.x' (errno: 1478)

mysql> **SHOW ERRORS;**

+-------+------+-------------------------------------------+

| Level | Code | Message |

+-------+------+-------------------------------------------+

| Error | 1478 | InnoDB: invalid KEY\_BLOCK\_SIZE=33333. |

| Error | 1005 | Can't create table 'test.x' (errno: 1478) |

+-------+------+-------------------------------------------+

**Table 15.13 CREATE/ALTER TABLE Warnings and Errors when InnoDB Strict Mode is OFF**

| **Syntax** | **Warning or Error Condition** | **Resulting ROW\_FORMAT, as shown in SHOW TABLE STATUS** |
| --- | --- | --- |
| **ROW\_FORMAT=REDUNDANT** | None | **REDUNDANT** |
| **ROW\_FORMAT=COMPACT** | None | **COMPACT** |
| **ROW\_FORMAT=COMPRESSED or ROW\_FORMAT=DYNAMIC or KEY\_BLOCK\_SIZE is specified** | Ignored for file-per-table tablespaces unless [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is enabled. General tablespaces support all row formats. See [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces). | **the default row format for file-per-table tablespaces; the specified row format for general tablespaces** |
| **Invalid KEY\_BLOCK\_SIZE is specified (not 1, 2, 4, 8 or 16)** | **KEY\_BLOCK\_SIZE** is ignored | the specified row format, or the default row format |
| **ROW\_FORMAT=COMPRESSED and valid KEY\_BLOCK\_SIZE are specified** | None; **KEY\_BLOCK\_SIZE** specified is used | **COMPRESSED** |
| **KEY\_BLOCK\_SIZE is specified with REDUNDANT, COMPACT or DYNAMIC row format** | **KEY\_BLOCK\_SIZE** is ignored | **REDUNDANT**, **COMPACT** or **DYNAMIC** |
| **ROW\_FORMAT is not one of REDUNDANT, COMPACT, DYNAMIC or COMPRESSED** | Ignored if recognized by the MySQL parser. Otherwise, an error is issued. | the default row format or N/A |

When **innodb\_strict\_mode** is **ON**, MySQL rejects invalid **ROW\_FORMAT** or **KEY\_BLOCK\_SIZE** parameters and issues errors. Strict mode is **ON** by default. When **innodb\_strict\_mode** is **OFF**, MySQL issues warnings instead of errors for ignored invalid parameters.

It is not possible to see the chosen **KEY\_BLOCK\_SIZE** using **SHOW TABLE STATUS**. The statement **SHOW CREATE TABLE** displays the **KEY\_BLOCK\_SIZE** (even if it was ignored when creating the table). The real compressed page size of the table cannot be displayed by MySQL.

##### SQL Compression Syntax Warnings and Errors for General Tablespaces

If **FILE\_BLOCK\_SIZE** was not defined for the general tablespace when the tablespace was created, the tablespace cannot contain compressed tables. If you attempt to add a compressed table, an error is returned, as shown in the following example:

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' Engine=InnoDB;**

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1 ROW\_FORMAT=COMPRESSED**

**KEY\_BLOCK\_SIZE=8;**

ERROR 1478 (HY000): InnoDB: Tablespace `ts1` cannot contain a COMPRESSED table

Attempting to add a table with an invalid **KEY\_BLOCK\_SIZE** to a general tablespace returns an error, as shown in the following example:

mysql> **CREATE TABLESPACE `ts2` ADD DATAFILE 'ts2.ibd' FILE\_BLOCK\_SIZE = 8192 Engine=InnoDB;**

mysql> **CREATE TABLE t2 (c1 INT PRIMARY KEY) TABLESPACE ts2 ROW\_FORMAT=COMPRESSED**

**KEY\_BLOCK\_SIZE=4;**

ERROR 1478 (HY000): InnoDB: Tablespace `ts2` uses block size 8192 and cannot

contain a table with physical page size 4096

For general tablespaces, the **KEY\_BLOCK\_SIZE** of the table must be equal to the **FILE\_BLOCK\_SIZE** of the tablespace divided by 1024. For example, if the **FILE\_BLOCK\_SIZE** of the tablespace is 8192, the **KEY\_BLOCK\_SIZE** of the table must be 8.

Attempting to add a table with an uncompressed row format to a general tablespace configured to store compressed tables returns an error, as shown in the following example:

mysql> **CREATE TABLESPACE `ts3` ADD DATAFILE 'ts3.ibd' FILE\_BLOCK\_SIZE = 8192 Engine=InnoDB;**

mysql> **CREATE TABLE t3 (c1 INT PRIMARY KEY) TABLESPACE ts3 ROW\_FORMAT=COMPACT;**

ERROR 1478 (HY000): InnoDB: Tablespace `ts3` uses block size 8192 and cannot

contain a table with physical page size 16384

[**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is not applicable to general tablespaces. Tablespace management rules for general tablespaces are strictly enforced independently of [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode). For more information, see [Section 13.1.21, “CREATE TABLESPACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace).

For more information about using compressed tables with general tablespaces, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

### 15.9.2 InnoDB Page Compression

**InnoDB** supports page-level compression for tables that reside in [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces. This feature is referred to as Transparent Page Compression. Page compression is enabled by specifying the **COMPRESSION** attribute with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Supported compression algorithms include **Zlib** and **LZ4**.

#### Supported Platforms

Page compression requires sparse file and hole punching support. Page compression is supported on Windows with NTFS, and on the following subset of MySQL-supported Linux platforms where the kernel level provides hole punching support:

RHEL 7 and derived distributions that use kernel version 3.10.0-123 or higher

OEL 5.10 (UEK2) kernel version 2.6.39 or higher

OEL 6.5 (UEK3) kernel version 3.8.13 or higher

OEL 7.0 kernel version 3.8.13 or higher

SLE11 kernel version 3.0-x

SLE12 kernel version 3.12-x

OES11 kernel version 3.0-x

Ubuntu 14.0.4 LTS kernel version 3.13 or higher

Ubuntu 12.0.4 LTS kernel version 3.2 or higher

Debian 7 kernel version 3.2 or higher

**Note**

All of the available file systems for a given Linux distribution may not support hole punching.

#### How Page Compression Works

When a page is written, it is compressed using the specified compression algorithm. The compressed data is written to disk, where the hole punching mechanism releases empty blocks from the end of the page. If compression fails, data is written out as-is.

#### Hole Punch Size on Linux

On Linux systems, the file system block size is the unit size used for hole punching. Therefore, page compression only works if page data can be compressed to a size that is less than or equal to the **InnoDB** page size minus the file system block size. For example, if [**innodb\_page\_size=16K**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) and the file system block size is 4K, page data must compress to less than or equal to 12K to make hole punching possible.

#### Hole Punch Size on Windows

On Windows systems, the underlying infrastructure for sparse files is based on NTFS compression. Hole punching size is the NTFS compression unit, which is 16 times the NTFS cluster size. Cluster sizes and their compression units are shown in the following table:

**Table 15.14 Windows NTFS Cluster Size and Compression Units**

| **Cluster Size** | **Compression Unit** |
| --- | --- |
| 512 Bytes | 8 KB |
| 1 KB | 16 KB |
| 2 KB | 32 KB |
| 4 KB | 64 KB |

Page compression on Windows systems only works if page data can be compressed to a size that is less than or equal to the **InnoDB** page size minus the compression unit size.

The default NTFS cluster size is 4KB, for which the compression unit size is 64KB. This means that page compression has no benefit for an out-of-the box Windows NTFS configuration, as the maximum [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is also 64KB.

For page compression to work on Windows, the file system must be created with a cluster size smaller than 4K, and the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) must be at least twice the size of the compression unit. For example, for page compression to work on Windows, you could build the file system with a cluster size of 512 Bytes (which has a compression unit of 8KB) and initialize **InnoDB** with an [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value of 16K or greater.

#### Enabling Page Compression

To enable page compression, specify the **COMPRESSION** attribute in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. For example:

CREATE TABLE t1 (c1 INT) COMPRESSION="zlib";

You can also enable page compression in an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. However, [**ALTER TABLE ... COMPRESSION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) only updates the tablespace compression attribute. Writes to the tablespace that occur after setting the new compression algorithm use the new setting, but to apply the new compression algorithm to existing pages, you must rebuild the table using [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table).

ALTER TABLE t1 COMPRESSION="zlib";

OPTIMIZE TABLE t1;

#### Disabling Page Compression

To disable page compression, set **COMPRESSION=None** using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Writes to the tablespace that occur after setting **COMPRESSION=None** no longer use page compression. To uncompress existing pages, you must rebuild the table using [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) after setting **COMPRESSION=None**.

ALTER TABLE t1 COMPRESSION="None";

OPTIMIZE TABLE t1;

#### Page Compression Metadata

Page compression metadata is found in the [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table, in the following columns:

**FS\_BLOCK\_SIZE**: The file system block size, which is the unit size used for hole punching.

**FILE\_SIZE**: The apparent size of the file, which represents the maximum size of the file, uncompressed.

**ALLOCATED\_SIZE**: The actual size of the file, which is the amount of space allocated on disk.

**Note**

On Unix-like systems, **ls -l *tablespace\_name*.ibd** shows the apparent file size (equivalent to **FILE\_SIZE**) in bytes. To view the actual amount of space allocated on disk (equivalent to **ALLOCATED\_SIZE**), use **du --block-size=1 *tablespace\_name*.ibd**. The **--block-size=1** option prints the allocated space in bytes instead of blocks, so that it can be compared to **ls -l** output.

Use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) to view the current page compression setting (**Zlib**, **Lz4**, or **None**). A table may contain a mix of pages with different compression settings.

In the following example, page compression metadata for the employees table is retrieved from the [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table.

# Create the employees table with Zlib page compression

CREATE TABLE employees (

emp\_no INT NOT NULL,

birth\_date DATE NOT NULL,

first\_name VARCHAR(14) NOT NULL,

last\_name VARCHAR(16) NOT NULL,

gender ENUM ('M','F') NOT NULL,

hire\_date DATE NOT NULL,

PRIMARY KEY (emp\_no)

) COMPRESSION="zlib";

# Insert data (not shown)

# Query page compression metadata in INFORMATION\_SCHEMA.INNODB\_TABLESPACES

mysql> **SELECT SPACE, NAME, FS\_BLOCK\_SIZE, FILE\_SIZE, ALLOCATED\_SIZE FROM**

**INFORMATION\_SCHEMA.INNODB\_TABLESPACES WHERE NAME='employees/employees'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 45

NAME: employees/employees

FS\_BLOCK\_SIZE: 4096

FILE\_SIZE: 23068672

ALLOCATED\_SIZE: 19415040

Page compression metadata for the employees table shows that the apparent file size is 23068672 bytes while the actual file size (with page compression) is 19415040 bytes. The file system block size is 4096 bytes, which is the block size used for hole punching.

#### Identifying Tables Using Page Compression

To identify tables for which page compression is enabled, you can query the [**INFORMATION\_SCHEMA.TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) **CREATE\_OPTIONS** column for tables defined with the **COMPRESSION** attribute:

mysql> **SELECT TABLE\_NAME, TABLE\_SCHEMA, CREATE\_OPTIONS FROM INFORMATION\_SCHEMA.TABLES**

**WHERE CREATE\_OPTIONS LIKE '%COMPRESSION=%';**

+------------+--------------+--------------------+

| TABLE\_NAME | TABLE\_SCHEMA | CREATE\_OPTIONS |

+------------+--------------+--------------------+

| employees | test | COMPRESSION="zlib" |

+------------+--------------+--------------------+

[**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) also shows the **COMPRESSION** attribute, if used.

#### Page Compression Limitations and Usage Notes

Page compression is disabled if the file system block size (or compression unit size on Windows) \* 2 > [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size).

Page compression is not supported for tables that reside in shared tablespaces, which include the system tablespace, temporary tablespaces, and general tablespaces.

Page compression is not supported for undo log tablespaces.

Page compression is not supported for redo log pages.

R-tree pages, which are used for spatial indexes, are not compressed.

Pages that belong to compressed tables (**ROW\_FORMAT=COMPRESSED**) are left as-is.

During recovery, updated pages are written out in an uncompressed form.

Loading a page-compressed tablespace on a server that does not support the compression algorithm that was used causes an I/O error.

Before downgrading to an earlier version of MySQL that does not support page compression, uncompress the tables that use the page compression feature. To uncompress a table, run [**ALTER TABLE ... COMPRESSION=None**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table).

Page-compressed tablespaces can be copied between Linux and Windows servers if the compression algorithm that was used is available on both servers.

Preserving page compression when moving a page-compressed tablespace file from one host to another requires a utility that preserves sparse files.

Better page compression may be achieved on Fusion-io hardware with NVMFS than on other platforms, as NVMFS is designed to take advantage of punch hole functionality.

Using the page compression feature with a large **InnoDB** page size and relatively small file system block size could result in write amplification. For example, a maximum **InnoDB** page size of 64KB with a 4KB file system block size may improve compression but may also increase demand on the buffer pool, leading to increased I/O and potential write amplification.

## 15.10 InnoDB Row Formats

The row format of a table determines how its rows are physically stored, which in turn can affect the performance of queries and DML operations. As more rows fit into a single disk page, queries and index lookups can work faster, less cache memory is required in the buffer pool, and less I/O is required to write out updated values.

The data in each table is divided into pages. The pages that make up each table are arranged in a tree data structure called a B-tree index. Table data and secondary indexes both use this type of structure. The B-tree index that represents an entire table is known as the clustered index, which is organized according to the primary key columns. The nodes of a clustered index data structure contain the values of all columns in the row. The nodes of a secondary index structure contain the values of index columns and primary key columns.

Variable-length columns are an exception to the rule that column values are stored in B-tree index nodes. Variable-length columns that are too long to fit on a B-tree page are stored on separately allocated disk pages called overflow pages. Such columns are referred to as off-page columns. The values of off-page columns are stored in singly-linked lists of overflow pages, with each such column having its own list of one or more overflow pages. Depending on column length, all or a prefix of variable-length column values are stored in the B-tree to avoid wasting storage and having to read a separate page.

The **InnoDB** storage engine supports four row formats: **REDUNDANT**, **COMPACT**, **DYNAMIC**, and **COMPRESSED**.

**Table 15.15 InnoDB Row Format Overview**

| **Row Format** | **Compact Storage Characteristics** | **Enhanced Variable-Length Column Storage** | **Large Index Key Prefix Support** | **Compression Support** | **Supported Tablespace Types** |
| --- | --- | --- | --- | --- | --- |
| **REDUNDANT** | No | No | No | No | system, file-per-table, general |
| **COMPACT** | Yes | No | No | No | system, file-per-table, general |
| **DYNAMIC** | Yes | Yes | Yes | No | system, file-per-table, general |
| **COMPRESSED** | Yes | Yes | Yes | Yes | file-per-table, general |

The topics that follow describe row format storage characteristics and how to define and determine the row format of a table.

[REDUNDANT Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-redundant)

[COMPACT Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-compact)

[DYNAMIC Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-dynamic)

[COMPRESSED Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-compressed)

[Defining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-defining)

[Determining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-detrmining)

### REDUNDANT Row Format

The **REDUNDANT** format provides compatibility with older versions of MySQL.

Tables that use the **REDUNDANT** row format store the first 768 bytes of variable-length column values ([**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types) in the index record within the B-tree node, with the remainder stored on overflow pages. Fixed-length columns greater than or equal to 768 bytes are encoded as variable-length columns, which can be stored off-page. For example, a **CHAR(255)** column can exceed 768 bytes if the maximum byte length of the character set is greater than 3, as it is with **utf8mb4**.

If the value of a column is 768 bytes or less, an overflow page is not used, and some savings in I/O may result, since the value is stored entirely in the B-tree node. This works well for relatively short **BLOB** column values, but may cause B-tree nodes to fill with data rather than key values, reducing their efficiency. Tables with many **BLOB** columns could cause B-tree nodes to become too full, and contain too few rows, making the entire index less efficient than if rows were shorter or column values were stored off-page.

#### REDUNDANT Row Format Storage Characteristics

The **REDUNDANT** row format has the following storage characteristics:

Each index record contains a 6-byte header. The header is used to link together consecutive records, and for row-level locking.

Records in the clustered index contain fields for all user-defined columns. In addition, there is a 6-byte transaction ID field and a 7-byte roll pointer field.

If no primary key is defined for a table, each clustered index record also contains a 6-byte row ID field.

Each secondary index record contains all the primary key columns defined for the clustered index key that are not in the secondary index.

A record contains a pointer to each field of the record. If the total length of the fields in a record is less than 128 bytes, the pointer is one byte; otherwise, two bytes. The array of pointers is called the record directory. The area where the pointers point is the data part of the record.

Internally, fixed-length character columns such as [**CHAR(10)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) in stored in fixed-length format. Trailing spaces are not truncated from [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns.

Fixed-length columns greater than or equal to 768 bytes are encoded as variable-length columns, which can be stored off-page. For example, a **CHAR(255)** column can exceed 768 bytes if the maximum byte length of the character set is greater than 3, as it is with **utf8mb4**.

An SQL **NULL** value reserves one or two bytes in the record directory. An SQL **NULL** value reserves zero bytes in the data part of the record if stored in a variable-length column. For a fixed-length column, the fixed length of the column is reserved in the data part of the record. Reserving fixed space for **NULL** values permits columns to be updated in place from **NULL** to non-**NULL** values without causing index page fragmentation.

### COMPACT Row Format

The **COMPACT** row format reduces row storage space by about 20% compared to the **REDUNDANT** row format, at the cost of increasing CPU use for some operations. If your workload is a typical one that is limited by cache hit rates and disk speed, **COMPACT** format is likely to be faster. If the workload is limited by CPU speed, compact format might be slower.

Tables that use the **COMPACT** row format store the first 768 bytes of variable-length column values ([**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types) in the index record within the [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) node, with the remainder stored on overflow pages. Fixed-length columns greater than or equal to 768 bytes are encoded as variable-length columns, which can be stored off-page. For example, a **CHAR(255)** column can exceed 768 bytes if the maximum byte length of the character set is greater than 3, as it is with **utf8mb4**.

If the value of a column is 768 bytes or less, an overflow page is not used, and some savings in I/O may result, since the value is stored entirely in the B-tree node. This works well for relatively short **BLOB** column values, but may cause B-tree nodes to fill with data rather than key values, reducing their efficiency. Tables with many **BLOB** columns could cause B-tree nodes to become too full, and contain too few rows, making the entire index less efficient than if rows were shorter or column values were stored off-page.

#### COMPACT Row Format Storage Characteristics

The **COMPACT** row format has the following storage characteristics:

Each index record contains a 5-byte header that may be preceded by a variable-length header. The header is used to link together consecutive records, and for row-level locking.

The variable-length part of the record header contains a bit vector for indicating **NULL** columns. If the number of columns in the index that can be **NULL** is ***N***, the bit vector occupies **CEILING(*N*/8)** bytes. (For example, if there are anywhere from 9 to 16 columns that can be **NULL**, the bit vector uses two bytes.) Columns that are **NULL** do not occupy space other than the bit in this vector. The variable-length part of the header also contains the lengths of variable-length columns. Each length takes one or two bytes, depending on the maximum length of the column. If all columns in the index are **NOT NULL** and have a fixed length, the record header has no variable-length part.

For each non-**NULL** variable-length field, the record header contains the length of the column in one or two bytes. Two bytes are only needed if part of the column is stored externally in overflow pages or the maximum length exceeds 255 bytes and the actual length exceeds 127 bytes. For an externally stored column, the 2-byte length indicates the length of the internally stored part plus the 20-byte pointer to the externally stored part. The internal part is 768 bytes, so the length is 768+20. The 20-byte pointer stores the true length of the column.

The record header is followed by the data contents of non-**NULL** columns.

Records in the clustered index contain fields for all user-defined columns. In addition, there is a 6-byte transaction ID field and a 7-byte roll pointer field.

If no primary key is defined for a table, each clustered index record also contains a 6-byte row ID field.

Each secondary index record contains all the primary key columns defined for the clustered index key that are not in the secondary index. If any of the primary key columns are variable length, the record header for each secondary index has a variable-length part to record their lengths, even if the secondary index is defined on fixed-length columns.

Internally, for nonvariable-length character sets, fixed-length character columns such as [**CHAR(10)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) are stored in a fixed-length format.

Trailing spaces are not truncated from [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns.

Internally, for variable-length character sets such as **utf8mb3** and **utf8mb4**, **InnoDB** attempts to store [**CHAR(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) in ***N*** bytes by trimming trailing spaces. If the byte length of a [**CHAR(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column value exceeds ***N*** bytes, trailing spaces are trimmed to a minimum of the column value byte length. The maximum length of a [**CHAR(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column is the maximum character byte length × ***N***.

A minimum of ***N*** bytes is reserved for [**CHAR(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char). Reserving the minimum space ***N*** in many cases enables column updates to be done in place without causing index page fragmentation. By comparison, [**CHAR(*N*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns occupy the maximum character byte length × ***N*** when using the **REDUNDANT** row format.

Fixed-length columns greater than or equal to 768 bytes are encoded as variable-length fields, which can be stored off-page. For example, a **CHAR(255)** column can exceed 768 bytes if the maximum byte length of the character set is greater than 3, as it is with **utf8mb4**.

### DYNAMIC Row Format

The **DYNAMIC** row format offers the same storage characteristics as the **COMPACT** row format but adds enhanced storage capabilities for long variable-length columns and supports large index key prefixes.

When a table is created with **ROW\_FORMAT=DYNAMIC**, **InnoDB** can store long variable-length column values (for [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types) fully off-page, with the clustered index record containing only a 20-byte pointer to the overflow page. Fixed-length fields greater than or equal to 768 bytes are encoded as variable-length fields. For example, a **CHAR(255)** column can exceed 768 bytes if the maximum byte length of the character set is greater than 3, as it is with **utf8mb4**.

Whether columns are stored off-page depends on the page size and the total size of the row. When a row is too long, the longest columns are chosen for off-page storage until the clustered index record fits on the [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) page. [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns that are less than or equal to 40 bytes are stored in line.

The **DYNAMIC** row format maintains the efficiency of storing the entire row in the index node if it fits (as do the **COMPACT** and **REDUNDANT** formats), but the **DYNAMIC** row format avoids the problem of filling B-tree nodes with a large number of data bytes of long columns. The **DYNAMIC** row format is based on the idea that if a portion of a long data value is stored off-page, it is usually most efficient to store the entire value off-page. With **DYNAMIC** format, shorter columns are likely to remain in the B-tree node, minimizing the number of overflow pages required for a given row.

The **DYNAMIC** row format supports index key prefixes up to 3072 bytes.

Tables that use the **DYNAMIC** row format can be stored in the system tablespace, file-per-table tablespaces, and general tablespaces. To store **DYNAMIC** tables in the system tablespace, either disable [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) and use a regular **CREATE TABLE** or **ALTER TABLE** statement, or use the **TABLESPACE [=] innodb\_system** table option with **CREATE TABLE** or **ALTER TABLE**. The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable is not applicable to general tablespaces, nor is it applicable when using the **TABLESPACE [=] innodb\_system** table option to store **DYNAMIC** tables in the system tablespace.

#### DYNAMIC Row Format Storage Characteristics

The **DYNAMIC** row format is a variation of the **COMPACT** row format. For storage characteristics, see [COMPACT Row Format Storage Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compact-row-format-characteristics).

### COMPRESSED Row Format

The **COMPRESSED** row format offers the same storage characteristics and capabilities as the **DYNAMIC** row format but adds support for table and index data compression.

The **COMPRESSED** row format uses similar internal details for off-page storage as the **DYNAMIC** row format, with additional storage and performance considerations from the table and index data being compressed and using smaller page sizes. With the **COMPRESSED** row format, the **KEY\_BLOCK\_SIZE** option controls how much column data is stored in the clustered index, and how much is placed on overflow pages. For more information about the **COMPRESSED** row format, see [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression).

The **COMPRESSED** row format supports index key prefixes up to 3072 bytes.

Tables that use the **COMPRESSED** row format can be created in file-per-table tablespaces or general tablespaces. The system tablespace does not support the **COMPRESSED** row format. To store a **COMPRESSED** table in a file-per-table tablespace, the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable must be enabled. The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable is not applicable to general tablespaces. General tablespaces support all row formats with the caveat that compressed and uncompressed tables cannot coexist in the same general tablespace due to different physical page sizes. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

#### Compressed Row Format Storage Characteristics

The **COMPRESSED** row format is a variation of the **COMPACT** row format. For storage characteristics, see [COMPACT Row Format Storage Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compact-row-format-characteristics).

### Defining the Row Format of a Table

The default row format for **InnoDB** tables is defined by [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable, which has a default value of **DYNAMIC**. The default row format is used when the **ROW\_FORMAT** table option is not defined explicitly or when **ROW\_FORMAT=DEFAULT** is specified.

The row format of a table can be defined explicitly using the **ROW\_FORMAT** table option in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. For example:

CREATE TABLE t1 (c1 INT) ROW\_FORMAT=DYNAMIC;

An explicitly defined **ROW\_FORMAT** setting overrides the default row format. Specifying **ROW\_FORMAT=DEFAULT** is equivalent to using the implicit default.

The [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable can be set dynamically:

mysql> **SET GLOBAL innodb\_default\_row\_format=DYNAMIC;**

Valid [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) options include **DYNAMIC**, **COMPACT**, and **REDUNDANT**. The **COMPRESSED** row format, which is not supported for use in the system tablespace, cannot be defined as the default. It can only be specified explicitly in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. Attempting to set the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable to **COMPRESSED** returns an error:

mysql> **SET GLOBAL innodb\_default\_row\_format=COMPRESSED;**

ERROR 1231 (42000): Variable 'innodb\_default\_row\_format'

can't be set to the value of 'COMPRESSED'

Newly created tables use the row format defined by the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable when a **ROW\_FORMAT** option is not specified explicitly, or when **ROW\_FORMAT=DEFAULT** is used. For example, the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements use the row format defined by the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable.

CREATE TABLE t1 (c1 INT);

CREATE TABLE t2 (c1 INT) ROW\_FORMAT=DEFAULT;

When a **ROW\_FORMAT** option is not specified explicitly, or when **ROW\_FORMAT=DEFAULT** is used, an operation that rebuilds a table silently changes the row format of the table to the format defined by the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable.

Table-rebuilding operations include [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that use **ALGORITHM=COPY** or **ALGORITHM=INPLACE** where table rebuilding is required. See [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations) for more information. [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is also a table-rebuilding operation.

The following example demonstrates a table-rebuilding operation that silently changes the row format of a table created without an explicitly defined row format.

mysql> **SELECT @@innodb\_default\_row\_format;**

+-----------------------------+

| @@innodb\_default\_row\_format |

+-----------------------------+

| dynamic |

+-----------------------------+

mysql> **CREATE TABLE t1 (c1 INT);**

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLES WHERE NAME LIKE 'test/t1' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 54

NAME: test/t1

FLAG: 33

N\_COLS: 4

SPACE: 35

ROW\_FORMAT: Dynamic

ZIP\_PAGE\_SIZE: 0

SPACE\_TYPE: Single

mysql> **SET GLOBAL innodb\_default\_row\_format=COMPACT;**

mysql> **ALTER TABLE t1 ADD COLUMN (c2 INT);**

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLES WHERE NAME LIKE 'test/t1' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 55

NAME: test/t1

FLAG: 1

N\_COLS: 5

SPACE: 36

ROW\_FORMAT: Compact

ZIP\_PAGE\_SIZE: 0

SPACE\_TYPE: Single

Consider the following potential issues before changing the row format of existing tables from **REDUNDANT** or **COMPACT** to **DYNAMIC**.

The **REDUNDANT** and **COMPACT** row formats support a maximum index key prefix length of 767 bytes whereas **DYNAMIC** and **COMPRESSED** row formats support an index key prefix length of 3072 bytes. In a replication environment, if the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) variable is set to **DYNAMIC** on the source, and set to **COMPACT** on the replica, the following DDL statement, which does not explicitly define a row format, succeeds on the source but fails on the replica:

CREATE TABLE t1 (c1 INT PRIMARY KEY, c2 VARCHAR(5000), KEY i1(c2(3070)));

For related information, see [Section 15.22, “InnoDB Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-limits).

Importing a table that does not explicitly define a row format results in a schema mismatch error if the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) setting on the source server differs from the setting on the destination server. For more information, see [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

### Determining the Row Format of a Table

To determine the row format of a table, use [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status):

mysql> **SHOW TABLE STATUS IN test1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: t1

Engine: InnoDB

Version: 10

Row\_format: Dynamic

Rows: 0

Avg\_row\_length: 0

Data\_length: 16384

Max\_data\_length: 0

Index\_length: 16384

Data\_free: 0

Auto\_increment: 1

Create\_time: 2016-09-14 16:29:38

Update\_time: NULL

Check\_time: NULL

Collation: utf8mb4\_0900\_ai\_ci

Checksum: NULL

Create\_options:

Comment:

Alternatively, query the [**INFORMATION\_SCHEMA.INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) table:

mysql> **SELECT NAME, ROW\_FORMAT FROM INFORMATION\_SCHEMA.INNODB\_TABLES WHERE NAME='test1/t1';**

+----------+------------+

| NAME | ROW\_FORMAT |

+----------+------------+

| test1/t1 | Dynamic |

+----------+------------+

## 15.11 InnoDB Disk I/O and File Space Management

[15.11.1 InnoDB Disk I/O](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disk-io)

[15.11.2 File Space Management](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-space)

[15.11.3 InnoDB Checkpoints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-checkpoints)

[15.11.4 Defragmenting a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-defragmenting)

[15.11.5 Reclaiming Disk Space with TRUNCATE TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-truncate-table-reclaim-space)

As a DBA, you must manage disk I/O to keep the I/O subsystem from becoming saturated, and manage disk space to avoid filling up storage devices. The [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid) design model requires a certain amount of I/O that might seem redundant, but helps to ensure data reliability. Within these constraints, **InnoDB** tries to optimize the database work and the organization of disk files to minimize the amount of disk I/O. Sometimes, I/O is postponed until the database is not busy, or until everything needs to be brought to a consistent state, such as during a database restart after a [fast shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fast_shutdown).

This section discusses the main considerations for I/O and disk space with the default kind of MySQL tables (also known as **InnoDB** tables):

Controlling the amount of background I/O used to improve query performance.

Enabling or disabling features that provide extra durability at the expense of additional I/O.

Organizing tables into many small files, a few larger files, or a combination of both.

Balancing the size of redo log files against the I/O activity that occurs when the log files become full.

How to reorganize a table for optimal query performance.

### 15.11.1 InnoDB Disk I/O

**InnoDB** uses asynchronous disk I/O where possible, by creating a number of threads to handle I/O operations, while permitting other database operations to proceed while the I/O is still in progress. On Linux and Windows platforms, **InnoDB** uses the available OS and library functions to perform “native” asynchronous I/O. On other platforms, **InnoDB** still uses I/O threads, but the threads may actually wait for I/O requests to complete; this technique is known as “simulated” asynchronous I/O.

#### Read-Ahead

If **InnoDB** can determine there is a high probability that data might be needed soon, it performs read-ahead operations to bring that data into the buffer pool so that it is available in memory. Making a few large read requests for contiguous data can be more efficient than making several small, spread-out requests. There are two read-ahead heuristics in **InnoDB**:

In sequential read-ahead, if **InnoDB** notices that the access pattern to a segment in the tablespace is sequential, it posts in advance a batch of reads of database pages to the I/O system.

In random read-ahead, if **InnoDB** notices that some area in a tablespace seems to be in the process of being fully read into the buffer pool, it posts the remaining reads to the I/O system.

For information about configuring read-ahead heuristics, see [Section 15.8.3.4, “Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead).

#### Doublewrite Buffer

**InnoDB** uses a novel file flush technique involving a structure called the [doublewrite buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_doublewrite_buffer), which is enabled by default in most cases ([**innodb\_doublewrite=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite)). It adds safety to recovery following an unexpected exit or power outage, and improves performance on most varieties of Unix by reducing the need for **fsync()** operations.

Before writing pages to a data file, **InnoDB** first writes them to a storage area called the doublewrite buffer. Only after the write and the flush to the doublewrite buffer has completed does **InnoDB** write the pages to their proper positions in the data file. If there is an operating system, storage subsystem, or unexpected [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process exit in the middle of a page write (causing a [torn page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_torn_page) condition), **InnoDB** can later find a good copy of the page from the doublewrite buffer during recovery.

For more information about the doublewrite buffer, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

### 15.11.2 File Space Management

The data files that you define in the configuration file using the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) configuration option form the **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace). The files are logically concatenated to form the system tablespace. There is no striping in use. You cannot define where within the system tablespace your tables are allocated. In a newly created system tablespace, **InnoDB** allocates space starting from the first data file.

To avoid the issues that come with storing all tables and indexes inside the system tablespace, you can enable the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) configuration option (the default), which stores each newly created table in a separate tablespace file (with extension **.ibd**). For tables stored this way, there is less fragmentation within the disk file, and when the table is truncated, the space is returned to the operating system rather than still being reserved by InnoDB within the system tablespace. For more information, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces).

You can also store tables in [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace). General tablespaces are shared tablespaces created using [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax. They can be created outside of the MySQL data directory, are capable of holding multiple tables, and support tables of all row formats. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

#### Pages, Extents, Segments, and Tablespaces

Each tablespace consists of database [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page). Every tablespace in a MySQL instance has the same [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size). By default, all tablespaces have a page size of 16KB; you can reduce the page size to 8KB or 4KB by specifying the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) option when you create the MySQL instance. You can also increase the page size to 32KB or 64KB. For more information, refer to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) documentation.

The pages are grouped into [extents](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_extent) of size 1MB for pages up to 16KB in size (64 consecutive 16KB pages, or 128 8KB pages, or 256 4KB pages). For a page size of 32KB, extent size is 2MB. For page size of 64KB, extent size is 4MB. The “files” inside a tablespace are called [segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_segment) in **InnoDB**. (These segments are different from the [rollback segment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback_segment), which actually contains many tablespace segments.)

When a segment grows inside the tablespace, **InnoDB** allocates the first 32 pages to it one at a time. After that, **InnoDB** starts to allocate whole extents to the segment. **InnoDB** can add up to 4 extents at a time to a large segment to ensure good sequentiality of data.

Two segments are allocated for each index in **InnoDB**. One is for nonleaf nodes of the [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree), the other is for the leaf nodes. Keeping the leaf nodes contiguous on disk enables better sequential I/O operations, because these leaf nodes contain the actual table data.

Some pages in the tablespace contain bitmaps of other pages, and therefore a few extents in an **InnoDB** tablespace cannot be allocated to segments as a whole, but only as individual pages.

When you ask for available free space in the tablespace by issuing a [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) statement, **InnoDB** reports the extents that are definitely free in the tablespace. **InnoDB** always reserves some extents for cleanup and other internal purposes; these reserved extents are not included in the free space.

When you delete data from a table, **InnoDB** contracts the corresponding B-tree indexes. Whether the freed space becomes available for other users depends on whether the pattern of deletes frees individual pages or extents to the tablespace. Dropping a table or deleting all rows from it is guaranteed to release the space to other users, but remember that deleted rows are physically removed only by the [purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge) operation, which happens automatically some time after they are no longer needed for transaction rollbacks or consistent reads. (See [Section 15.3, “InnoDB Multi-Versioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multi-versioning).)

#### How Pages Relate to Table Rows

The maximum row length is slightly less than half a database page for 4KB, 8KB, 16KB, and 32KB [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) settings. For example, the maximum row length is slightly less than 8KB for the default 16KB **InnoDB** page size. For 64KB pages, the maximum row length is slightly less than 16KB.

If a row does not exceed the maximum row length, all of it is stored locally within the page. If a row exceeds the maximum row length, [variable-length columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_variable_length_type) are chosen for external off-page storage until the row fits within the maximum row length limit. External off-page storage for variable-length columns differs by row format:

COMPACT and REDUNDANT Row Formats

When a variable-length column is chosen for external off-page storage, **InnoDB** stores the first 768 bytes locally in the row, and the rest externally into overflow pages. Each such column has its own list of overflow pages. The 768-byte prefix is accompanied by a 20-byte value that stores the true length of the column and points into the overflow list where the rest of the value is stored. See [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

DYNAMIC and COMPRESSED Row Formats

When a variable-length column is chosen for external off-page storage, **InnoDB** stores a 20-byte pointer locally in the row, and the rest externally into overflow pages. See [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

[**LONGBLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**LONGTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns must be less than 4GB, and the total row length, including [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, must be less than 4GB.

### 15.11.3 InnoDB Checkpoints

Making your [log files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file) very large may reduce disk I/O during [checkpointing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint). It often makes sense to set the total size of the log files as large as the buffer pool or even larger.

#### How Checkpoint Processing Works

**InnoDB** implements a [checkpoint](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint) mechanism known as [fuzzy checkpointing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fuzzy_checkpointing). **InnoDB** flushes modified database pages from the buffer pool in small batches. There is no need to flush the buffer pool in one single batch, which would disrupt processing of user SQL statements during the checkpointing process.

During [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery), **InnoDB** looks for a checkpoint label written to the log files. It knows that all modifications to the database before the label are present in the disk image of the database. Then **InnoDB** scans the log files forward from the checkpoint, applying the logged modifications to the database.

### 15.11.4 Defragmenting a Table

Random insertions into or deletions from a secondary index can cause the index to become fragmented. Fragmentation means that the physical ordering of the index pages on the disk is not close to the index ordering of the records on the pages, or that there are many unused pages in the 64-page blocks that were allocated to the index.

One symptom of fragmentation is that a table takes more space than it “should” take. How much that is exactly, is difficult to determine. All **InnoDB** data and indexes are stored in [B-trees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree), and their [fill factor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fill_factor) may vary from 50% to 100%. Another symptom of fragmentation is that a table scan such as this takes more time than it “should” take:

SELECT COUNT(\*) FROM t WHERE ***non\_indexed\_column*** <> 12345;

The preceding query requires MySQL to perform a full table scan, the slowest type of query for a large table.

To speed up index scans, you can periodically perform a “null” [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation, which causes MySQL to rebuild the table:

ALTER TABLE ***tbl\_name*** ENGINE=INNODB

You can also use [**ALTER TABLE *tbl\_name* FORCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to perform a “null” alter operation that rebuilds the table.

Both [**ALTER TABLE *tbl\_name* ENGINE=INNODB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**ALTER TABLE *tbl\_name* FORCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) use [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl). For more information, see [Section 15.12, “InnoDB and Online DDL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl).

Another way to perform a defragmentation operation is to use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to dump the table to a text file, drop the table, and reload it from the dump file.

If the insertions into an index are always ascending and records are deleted only from the end, the **InnoDB** filespace management algorithm guarantees that fragmentation in the index does not occur.

### 15.11.5 Reclaiming Disk Space with TRUNCATE TABLE

To reclaim operating system disk space when [truncating](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_truncate) an **InnoDB** table, the table must be stored in its own [.ibd](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) file. For a table to be stored in its own [.ibd](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) file, [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) must enabled when the table is created. Additionally, there cannot be a [foreign key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_foreign_key) constraint between the table being truncated and other tables, otherwise the **TRUNCATE TABLE** operation fails. A foreign key constraint between two columns in the same table, however, is permitted.

When a table is truncated, it is dropped and re-created in a new .ibd file, and the freed space is returned to the operating system. This is in contrast to truncating **InnoDB** tables that are stored within the **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) (tables created when **innodb\_file\_per\_table=OFF**) and tables stored in shared [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace), where only **InnoDB** can use the freed space after the table is truncated.

The ability to truncate tables and return disk space to the operating system also means that [physical backups](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_physical_backup) can be smaller. Truncating tables that are stored in the system tablespace (tables created when **innodb\_file\_per\_table=OFF**) or in a general tablespace leaves blocks of unused space in the tablespace.

## 15.12 InnoDB and Online DDL

[15.12.1 Online DDL Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations)

[15.12.2 Online DDL Performance and Concurrency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-performance)

[15.12.3 Online DDL Space Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-space-requirements)

[15.12.4 Simplifying DDL Statements with Online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-single-multi)

[15.12.5 Online DDL Failure Conditions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-failure-conditions)

[15.12.6 Online DDL Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-limitations)

The online DDL feature provides support for instant and in-place table alterations and concurrent DML. Benefits of this feature include:

Improved responsiveness and availability in busy production environments, where making a table unavailable for minutes or hours is not practical.

For in-place operations, the ability to adjust the balance between performance and concurrency during DDL operations using the **LOCK** clause. See [The LOCK clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-locking-options).

Less disk space usage and I/O overhead than the table-copy method.

**Note**

**ALGORITHM=INSTANT** support is available for **ADD COLUMN** and other operations in MySQL 8.0.12.

Typically, you do not need to do anything special to enable online DDL. By default, MySQL performs the operation instantly or in place, as permitted, with as little locking as possible.

You can control aspects of a DDL operation using the **ALGORITHM** and **LOCK** clauses of the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. These clauses are placed at the end of the statement, separated from the table and column specifications by commas. For example:

ALTER TABLE ***tbl\_name*** ADD PRIMARY KEY (***column***), ALGORITHM=INPLACE, LOCK=NONE;

The **LOCK** clause may be used for operations that are performed in place and is useful for fine-tuning the degree of concurrent access to the table during operations. Only **LOCK=DEFAULT** is supported for operations that are performed instantly. The **ALGORITHM** clause is primarily intended for performance comparisons and as a fallback to the older table-copying behavior in case you encounter any issues. For example:

To avoid accidentally making the table unavailable for reads, writes, or both, during an in-place [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation, specify a clause on the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement such as **LOCK=NONE** (permit reads and writes) or **LOCK=SHARED** (permit reads). The operation halts immediately if the requested level of concurrency is not available.

To compare performance between algorithms, run a statement with **ALGORITHM=INSTANT**, **ALGORITHM=INPLACE** and **ALGORITHM=COPY**. You can also run a statement with the [**old\_alter\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table) configuration option enabled to force the use of **ALGORITHM=COPY**.

To avoid tying up the server with an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation that copies the table, include **ALGORITHM=INSTANT** or **ALGORITHM=INPLACE**. The statement halts immediately if it cannot use the specified algorithm.

### 15.12.1 Online DDL Operations

Online support details, syntax examples, and usage notes for DDL operations are provided under the following topics in this section.

[Index Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-index-operations)

[Primary Key Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-primary-key-operations)

[Column Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-column-operations)

[Generated Column Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-generated-column-operations)

[Foreign Key Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-foreign-key-operations)

[Table Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-table-operations)

[Tablespace Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-tablespace-operations)

[Partitioning Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-partitioning)

#### Index Operations

The following table provides an overview of online DDL support for index operations. An asterisk indicates additional information, an exception, or a dependency. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-index-syntax-notes).

**Table 15.16 Online DDL Support for Index Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Creating or adding a secondary index** | No | Yes | No | Yes | No |
| **Dropping an index** | No | Yes | No | Yes | Yes |
| **Renaming an index** | No | Yes | No | Yes | Yes |
| **Adding a FULLTEXT index** | No | Yes\* | No\* | No | No |
| **Adding a SPATIAL index** | No | Yes | No | No | No |
| **Changing the index type** | Yes | Yes | No | Yes | Yes |

##### Syntax and Usage Notes

Creating or adding a secondary index

CREATE INDEX ***name*** ON ***table*** (***col\_list***);

ALTER TABLE ***tbl\_name*** ADD INDEX ***name*** (***col\_list***);

The table remains available for read and write operations while the index is being created. The [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statement only finishes after all transactions that are accessing the table are completed, so that the initial state of the index reflects the most recent contents of the table.

Online DDL support for adding secondary indexes means that you can generally speed the overall process of creating and loading a table and associated indexes by creating the table without secondary indexes, then adding secondary indexes after the data is loaded.

A newly created secondary index contains only the committed data in the table at the time the [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement finishes executing. It does not contain any uncommitted values, old versions of values, or values marked for deletion but not yet removed from the old index.

Some factors affect the performance, space usage, and semantics of this operation. For details, see [Section 15.12.6, “Online DDL Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-limitations).

Dropping an index

DROP INDEX ***name*** ON ***table***;

ALTER TABLE ***tbl\_name*** DROP INDEX ***name***;

The table remains available for read and write operations while the index is being dropped. The [**DROP INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-index) statement only finishes after all transactions that are accessing the table are completed, so that the initial state of the index reflects the most recent contents of the table.

Renaming an index

ALTER TABLE ***tbl\_name*** RENAME INDEX ***old\_index\_name*** TO ***new\_index\_name***, ALGORITHM=INPLACE, LOCK=NONE;

Adding a **FULLTEXT** index

CREATE FULLTEXT INDEX ***name*** ON table(***column***);

Adding the first **FULLTEXT** index rebuilds the table if there is no user-defined **FTS\_DOC\_ID** column. Additional **FULLTEXT** indexes may be added without rebuilding the table.

Adding a **SPATIAL** index

CREATE TABLE geom (g GEOMETRY NOT NULL);

ALTER TABLE geom ADD SPATIAL INDEX(g), ALGORITHM=INPLACE, LOCK=SHARED;

Changing the index type (**USING {BTREE | HASH}**)

ALTER TABLE ***tbl\_name*** DROP INDEX i1, ADD INDEX i1(***key\_part,...***) USING BTREE, ALGORITHM=INSTANT;

#### Primary Key Operations

The following table provides an overview of online DDL support for primary key operations. An asterisk indicates additional information, an exception, or a dependency. See [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-primary-key-syntax-notes).

**Table 15.17 Online DDL Support for Primary Key Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Adding a primary key** | No | Yes\* | Yes\* | Yes | No |
| **Dropping a primary key** | No | No | Yes | No | No |
| **Dropping a primary key and adding another** | No | Yes | Yes | Yes | No |

##### Syntax and Usage Notes

Adding a primary key

ALTER TABLE ***tbl\_name*** ADD PRIMARY KEY (***column***), ALGORITHM=INPLACE, LOCK=NONE;

Rebuilds the table in place. Data is reorganized substantially, making it an expensive operation. **ALGORITHM=INPLACE** is not permitted under certain conditions if columns have to be converted to **NOT NULL**.

Restructuring the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index) always requires copying of table data. Thus, it is best to define the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) when you create a table, rather than issuing **ALTER TABLE ... ADD PRIMARY KEY** later.

When you create a **UNIQUE** or **PRIMARY KEY** index, MySQL must do some extra work. For **UNIQUE** indexes, MySQL checks that the table contains no duplicate values for the key. For a **PRIMARY KEY** index, MySQL also checks that none of the **PRIMARY KEY** columns contains a **NULL**.

When you add a primary key using the **ALGORITHM=COPY** clause, MySQL converts **NULL** values in the associated columns to default values: 0 for numbers, an empty string for character-based columns and BLOBs, and 0000-00-00 00:00:00 for **DATETIME**. This is a non-standard behavior that Oracle recommends you not rely on. Adding a primary key using **ALGORITHM=INPLACE** is only permitted when the [**SQL\_MODE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) setting includes the **strict\_trans\_tables** or **strict\_all\_tables** flags; when the **SQL\_MODE** setting is strict, **ALGORITHM=INPLACE** is permitted, but the statement can still fail if the requested primary key columns contain **NULL** values. The **ALGORITHM=INPLACE** behavior is more standard-compliant.

If you create a table without a primary key, **InnoDB** chooses one for you, which can be the first **UNIQUE** key defined on **NOT NULL** columns, or a system-generated key. To avoid uncertainty and the potential space requirement for an extra hidden column, specify the **PRIMARY KEY** clause as part of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

MySQL creates a new clustered index by copying the existing data from the original table to a temporary table that has the desired index structure. Once the data is completely copied to the temporary table, the original table is renamed with a different temporary table name. The temporary table comprising the new clustered index is renamed with the name of the original table, and the original table is dropped from the database.

The online performance enhancements that apply to operations on secondary indexes do not apply to the primary key index. The rows of an InnoDB table are stored in a [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index) organized based on the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key), forming what some database systems call an “index-organized table”. Because the table structure is closely tied to the primary key, redefining the primary key still requires copying the data.

When an operation on the primary key uses **ALGORITHM=INPLACE**, even though the data is still copied, it is more efficient than using **ALGORITHM=COPY** because:

No undo logging or associated redo logging is required for **ALGORITHM=INPLACE**. These operations add overhead to DDL statements that use **ALGORITHM=COPY**.

The secondary index entries are pre-sorted, and so can be loaded in order.

The change buffer is not used, because there are no random-access inserts into the secondary indexes.

Dropping a primary key

ALTER TABLE ***tbl\_name*** DROP PRIMARY KEY, ALGORITHM=COPY;

Only **ALGORITHM=COPY** supports dropping a primary key without adding a new one in the same **ALTER TABLE** statement.

Dropping a primary key and adding another

ALTER TABLE ***tbl\_name*** DROP PRIMARY KEY, ADD PRIMARY KEY (***column***), ALGORITHM=INPLACE, LOCK=NONE;

Data is reorganized substantially, making it an expensive operation.

#### Column Operations

The following table provides an overview of online DDL support for column operations. An asterisk indicates additional information, an exception, or a dependency. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-column-syntax-notes).

**Table 15.18 Online DDL Support for Column Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Adding a column** | Yes\* | Yes | No\* | Yes\* | No |
| **Dropping a column** | No | Yes | Yes | Yes | No |
| **Renaming a column** | No | Yes | No | Yes\* | Yes |
| **Reordering columns** | No | Yes | Yes | Yes | No |
| **Setting a column default value** | Yes | Yes | No | Yes | Yes |
| **Changing the column data type** | No | No | Yes | No | No |
| **Extending VARCHAR column size** | No | Yes | No | Yes | Yes |
| **Dropping the column default value** | Yes | Yes | No | Yes | Yes |
| **Changing the auto-increment value** | No | Yes | No | Yes | No\* |
| **Making a column NULL** | No | Yes | Yes\* | Yes | No |
| **Making a column NOT NULL** | No | Yes\* | Yes\* | Yes | No |
| **Modifying the definition of an ENUM or SET column** | Yes | Yes | No | Yes | Yes |

##### Syntax and Usage Notes

Adding a column

ALTER TABLE ***tbl\_name*** ADD COLUMN ***column\_name*** ***column\_definition***, ALGORITHM=INSTANT;

The following limitations apply when the **INSTANT** algorithm is used to add a column:

Adding a column cannot be combined in the same statement with other **ALTER TABLE** actions that do not support **ALGORITHM=INSTANT**.

A column can only be added as the last column of the table. Adding a column to any other position among other columns is not supported.

Columns cannot be added to tables that use **ROW\_FORMAT=COMPRESSED**.

Columns cannot be added to tables that include a **FULLTEXT** index.

Columns cannot be added to temporary tables. Temporary tables only support **ALGORITHM=COPY**.

Columns cannot be added to tables that reside in the data dictionary tablespace.

Row size limits are not evaluated when adding a column. However, row size limits are checked during DML operations that insert and update rows in the table.

Multiple columns may be added in the same [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. For example:

ALTER TABLE t1 ADD COLUMN c2 INT, ADD COLUMN c3 INT, ALGORITHM=INSTANT;

[**INFORMATION\_SCHEMA.INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) and [**INFORMATION\_SCHEMA.INNODB\_COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-columns-table) provide metadata for instantly added columns. **INFORMATION\_SCHEMA.INNODB\_TABLES.INSTANT\_COLS** shows number of columns in the table prior to adding the first instant column. **INFORMATION\_SCHEMA.INNODB\_COLUMNS.HAS\_DEFAULT** and **DEFAULT\_VALUE** provide metadata about default values for instantly added columns.

Concurrent DML is not permitted when adding an [auto-increment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_auto_increment) column. Data is reorganized substantially, making it an expensive operation. At a minimum, **ALGORITHM=INPLACE, LOCK=SHARED** is required.

The table is rebuilt if **ALGORITHM=INPLACE** is used to add a column.

Dropping a column

ALTER TABLE ***tbl\_name*** DROP COLUMN ***column\_name***, ALGORITHM=INPLACE, LOCK=NONE;

Data is reorganized substantially, making it an expensive operation.

Renaming a column

ALTER TABLE ***tbl*** CHANGE ***old\_col\_name*** ***new\_col\_name*** ***data\_type***, ALGORITHM=INPLACE, LOCK=NONE;

To permit concurrent DML, keep the same data type and only change the column name.

When you keep the same data type and **[NOT] NULL** attribute, only changing the column name, the operation can always be performed online.

You can also rename a column that is part of a foreign key constraint. The foreign key definition is automatically updated to use the new column name. Renaming a column participating in a foreign key only works with **ALGORITHM=INPLACE**. If you use the **ALGORITHM=COPY** clause, or some other condition causes the command to use **ALGORITHM=COPY** behind the scenes, the **ALTER TABLE** statement fails.

**ALGORITHM=INPLACE** is not supported for renaming a [generated column](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_generated_column).

Reordering columns

To reorder columns, use **FIRST** or **AFTER** in **CHANGE** or **MODIFY** operations.

ALTER TABLE ***tbl\_name*** MODIFY COLUMN ***col\_name*** ***column\_definition*** FIRST, ALGORITHM=INPLACE, LOCK=NONE;

Data is reorganized substantially, making it an expensive operation.

Changing the column data type

ALTER TABLE ***tbl\_name*** CHANGE c1 c1 BIGINT, ALGORITHM=COPY;

Changing the column data type is only supported with **ALGORITHM=COPY**.

Extending **VARCHAR** column size

ALTER TABLE ***tbl\_name*** CHANGE COLUMN c1 c1 VARCHAR(255), ALGORITHM=INPLACE, LOCK=NONE;

The number of length bytes required by a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column must remain the same. For [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns of 0 to 255 bytes in size, one length byte is required to encode the value. For [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns of 256 bytes in size or more, two length bytes are required. As a result, in-place [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) only supports increasing [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column size from 0 to 255 bytes, or from 256 bytes to a greater size. In-place [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) does not support increasing the size of a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column from less than 256 bytes to a size equal to or greater than 256 bytes. In this case, the number of required length bytes changes from 1 to 2, which is only supported by a table copy (**ALGORITHM=COPY**). For example, attempting to change [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column size for a single byte character set from VARCHAR(255) to VARCHAR(256) using in-place [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) returns this error:

ALTER TABLE ***tbl\_name*** ALGORITHM=INPLACE, CHANGE COLUMN c1 c1 VARCHAR(256);

ERROR 0A000: ALGORITHM=INPLACE is not supported. Reason: Cannot change

column type INPLACE. Try ALGORITHM=COPY.

**Note**

The byte length of a **VARCHAR** column is dependant on the byte length of the character set.

Decreasing [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) size using in-place [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is not supported. Decreasing [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) size requires a table copy (**ALGORITHM=COPY**).

Setting a column default value

ALTER TABLE ***tbl\_name*** ALTER COLUMN ***col*** SET DEFAULT ***literal***, ALGORITHM=INSTANT;

Only modifies table metadata. Default column values are stored in the [data dictionary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_dictionary).

Dropping a column default value

ALTER TABLE ***tbl*** ALTER COLUMN ***col*** DROP DEFAULT, ALGORITHM=INSTANT;

Changing the auto-increment value

ALTER TABLE ***table*** AUTO\_INCREMENT=***next\_value***, ALGORITHM=INPLACE, LOCK=NONE;

Modifies a value stored in memory, not the data file.

In a distributed system using replication or sharding, you sometimes reset the auto-increment counter for a table to a specific value. The next row inserted into the table uses the specified value for its auto-increment column. You might also use this technique in a data warehousing environment where you periodically empty all the tables and reload them, and restart the auto-increment sequence from 1.

Making a column **NULL**

ALTER TABLE tbl\_name MODIFY COLUMN ***column\_name*** ***data\_type*** NULL, ALGORITHM=INPLACE, LOCK=NONE;

Rebuilds the table in place. Data is reorganized substantially, making it an expensive operation.

Making a column **NOT NULL**

ALTER TABLE ***tbl\_name*** MODIFY COLUMN ***column\_name*** ***data\_type*** NOT NULL, ALGORITHM=INPLACE, LOCK=NONE;

Rebuilds the table in place. **STRICT\_ALL\_TABLES** or **STRICT\_TRANS\_TABLES** [**SQL\_MODE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) is required for the operation to succeed. The operation fails if the column contains NULL values. The server prohibits changes to foreign key columns that have the potential to cause loss of referential integrity. See [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Data is reorganized substantially, making it an expensive operation.

Modifying the definition of an **ENUM** or **SET** column

CREATE TABLE t1 (c1 ENUM('a', 'b', 'c'));

ALTER TABLE t1 MODIFY COLUMN c1 ENUM('a', 'b', 'c', 'd'), ALGORITHM=INSTANT;

Modifying the definition of an [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) column by adding new enumeration or set members to the end of the list of valid member values may be performed instantly or in place, as long as the storage size of the data type does not change. For example, adding a member to a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) column that has 8 members changes the required storage per value from 1 byte to 2 bytes; this requires a table copy. Adding members in the middle of the list causes renumbering of existing members, which requires a table copy.

#### Generated Column Operations

The following table provides an overview of online DDL support for generated column operations. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-generated-column-syntax-notes).

**Table 15.19 Online DDL Support for Generated Column Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Adding a STORED column** | No | No | Yes | No | No |
| **Modifying STORED column order** | No | No | Yes | No | No |
| **Dropping a STORED column** | No | Yes | Yes | Yes | No |
| **Adding a VIRTUAL column** | Yes | Yes | No | Yes | Yes |
| **Modifying VIRTUAL column order** | No | No | Yes | No | No |
| **Dropping a VIRTUAL column** | Yes | Yes | No | Yes | Yes |

##### Syntax and Usage Notes

Adding a **STORED** column

ALTER TABLE t1 ADD COLUMN (c2 INT GENERATED ALWAYS AS (c1 + 1) STORED), ALGORITHM=COPY;

**ADD COLUMN** is not an in-place operation for stored columns (done without using a temporary table) because the expression must be evaluated by the server.

Modifying **STORED** column order

ALTER TABLE t1 MODIFY COLUMN c2 INT GENERATED ALWAYS AS (c1 + 1) STORED FIRST, ALGORITHM=COPY;

Rebuilds the table in place.

Dropping a **STORED** column

ALTER TABLE t1 DROP COLUMN c2, ALGORITHM=INPLACE, LOCK=NONE;

Rebuilds the table in place.

Adding a **VIRTUAL** column

ALTER TABLE t1 ADD COLUMN (c2 INT GENERATED ALWAYS AS (c1 + 1) VIRTUAL), ALGORITHM=INSTANT;

Adding a virtual column can be performed instantly or in place for non-partitioned tables.

Adding a **VIRTUAL** is not an in-place operation for partitioned tables.

Modifying **VIRTUAL** column order

ALTER TABLE t1 MODIFY COLUMN c2 INT GENERATED ALWAYS AS (c1 + 1) VIRTUAL FIRST, ALGORITHM=COPY;

Dropping a **VIRTUAL** column

ALTER TABLE t1 DROP COLUMN c2, ALGORITHM=INSTANT;

Dropping a **VIRTUAL** column can be performed instantly or in place for non-partitioned tables.

#### Foreign Key Operations

The following table provides an overview of online DDL support for foreign key operations. An asterisk indicates additional information, an exception, or a dependency. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-foreign-key-syntax-notes).

**Table 15.20 Online DDL Support for Foreign Key Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Adding a foreign key constraint** | No | Yes\* | No | Yes | Yes |
| **Dropping a foreign key constraint** | No | Yes | No | Yes | Yes |

##### Syntax and Usage Notes

Adding a foreign key constraint

The **INPLACE** algorithm is supported when [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) is disabled. Otherwise, only the **COPY** algorithm is supported.

ALTER TABLE ***tbl1*** ADD CONSTRAINT ***fk\_name*** FOREIGN KEY ***index*** (***col1***)

REFERENCES ***tbl2***(***col2***) ***referential\_actions***;

Dropping a foreign key constraint

ALTER TABLE ***tbl*** DROP FOREIGN KEY ***fk\_name***;

Dropping a foreign key can be performed online with the [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) option enabled or disabled.

If you do not know the names of the foreign key constraints on a particular table, issue the following statement and find the constraint name in the **CONSTRAINT** clause for each foreign key:

SHOW CREATE TABLE ***table***\G

Or, query the [**INFORMATION\_SCHEMA.TABLE\_CONSTRAINTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-table-constraints-table) table and use the **CONSTRAINT\_NAME** and **CONSTRAINT\_TYPE** columns to identify the foreign key names.

You can also drop a foreign key and its associated index in a single statement:

ALTER TABLE ***table*** DROP FOREIGN KEY ***constraint***, DROP INDEX ***index***;

**Note**

If [foreign keys](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_foreign_key) are already present in the table being altered (that is, it is a [child table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_child_table) containing a **FOREIGN KEY ... REFERENCE** clause), additional restrictions apply to online DDL operations, even those not directly involving the foreign key columns:

An [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on the child table could wait for another transaction to commit, if a change to the parent table causes associated changes in the child table through an **ON UPDATE** or **ON DELETE** clause using the **CASCADE** or **SET NULL** parameters.

In the same way, if a table is the [parent table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_parent_table) in a foreign key relationship, even though it does not contain any **FOREIGN KEY** clauses, it could wait for the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to complete if an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement causes an **ON UPDATE** or **ON DELETE** action in the child table.

#### Table Operations

The following table provides an overview of online DDL support for table operations. An asterisk indicates additional information, an exception, or a dependency. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-table-syntax-notes).

**Table 15.21 Online DDL Support for Table Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Changing the ROW\_FORMAT** | No | Yes | Yes | Yes | No |
| **Changing the KEY\_BLOCK\_SIZE** | No | Yes | Yes | Yes | No |
| **Setting persistent table statistics** | No | Yes | No | Yes | Yes |
| **Specifying a character set** | No | Yes | Yes\* | No | No |
| **Converting a character set** | No | No | Yes\* | No | No |
| **Optimizing a table** | No | Yes\* | Yes | Yes | No |
| **Rebuilding with the FORCE option** | No | Yes\* | Yes | Yes | No |
| **Performing a null rebuild** | No | Yes\* | Yes | Yes | No |
| **Renaming a table** | Yes | Yes | No | Yes | Yes |

##### Syntax and Usage Notes

Changing the **ROW\_FORMAT**

ALTER TABLE ***tbl\_name*** ROW\_FORMAT = ***row\_format***, ALGORITHM=INPLACE, LOCK=NONE;

Data is reorganized substantially, making it an expensive operation.

For additional information about the **ROW\_FORMAT** option, see [Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-options).

Changing the **KEY\_BLOCK\_SIZE**

ALTER TABLE ***tbl\_name*** KEY\_BLOCK\_SIZE = ***value***, ALGORITHM=INPLACE, LOCK=NONE;

Data is reorganized substantially, making it an expensive operation.

For additional information about the **KEY\_BLOCK\_SIZE** option, see [Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-options).

Setting persistent table statistics options

ALTER TABLE ***tbl\_name*** STATS\_PERSISTENT=0, STATS\_SAMPLE\_PAGES=20, STATS\_AUTO\_RECALC=1, ALGORITHM=INPLACE, LOCK=NONE;

Only modifies table metadata.

Persistent statistics include **STATS\_PERSISTENT**, **STATS\_AUTO\_RECALC**, and **STATS\_SAMPLE\_PAGES**. For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

Specifying a character set

ALTER TABLE ***tbl\_name*** CHARACTER SET = ***charset\_name***, ALGORITHM=INPLACE, LOCK=NONE;

Rebuilds the table if the new character encoding is different.

Converting a character set

ALTER TABLE ***tbl\_name*** CONVERT TO CHARACTER SET ***charset\_name***, ALGORITHM=COPY;

Rebuilds the table if the new character encoding is different.

Optimizing a table

OPTIMIZE TABLE ***tbl\_name***;

In-place operation is not supported for tables with **FULLTEXT** indexes. The operation uses the **INPLACE** algorithm, but **ALGORITHM** and **LOCK** syntax is not permitted.

Rebuilding a table with the **FORCE** option

ALTER TABLE ***tbl\_name*** FORCE, ALGORITHM=INPLACE, LOCK=NONE;

Uses **ALGORITHM=INPLACE** as of MySQL 5.6.17. **ALGORITHM=INPLACE** is not supported for tables with **FULLTEXT** indexes.

Performing a "null" rebuild

ALTER TABLE ***tbl\_name*** ENGINE=InnoDB, ALGORITHM=INPLACE, LOCK=NONE;

Uses **ALGORITHM=INPLACE** as of MySQL 5.6.17. **ALGORITHM=INPLACE** is not supported for tables with **FULLTEXT** indexes.

Renaming a table

ALTER TABLE ***old\_tbl\_name*** RENAME TO ***new\_tbl\_name***, ALGORITHM=INSTANT;

Renaming a table can be performed instantly or in place. MySQL renames files that correspond to the table ***tbl\_name*** without making a copy. (You can also use the [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) statement to rename tables. See [Section 13.1.36, “RENAME TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table).) Privileges granted specifically for the renamed table are not migrated to the new name. They must be changed manually.

#### Tablespace Operations

The following table provides an overview of online DDL support for tablespace operations. For details, see [Syntax and Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#online-ddl-tablespace-syntax-notes).

**Table 15.22 Online DDL Support for Tablespace Operations**

| **Operation** | **Instant** | **In Place** | **Rebuilds Table** | **Permits Concurrent DML** | **Only Modifies Metadata** |
| --- | --- | --- | --- | --- | --- |
| **Renaming a general tablespace** | No | Yes | No | Yes | Yes |
| **Enabling or disabling general tablespace encryption** | No | Yes | No | Yes | No |
| **Enabling or disabling file-per-table tablespace encryption** | No | No | Yes | No | No |

##### Syntax and Usage Notes

Renaming a general tablespace

ALTER TABLESPACE ***tablespace\_name*** RENAME TO ***new\_tablespace\_name***;

[**ALTER TABLESPACE ... RENAME TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) uses the **INPLACE** algorithm but does not support the **ALGORITHM** clause.

Enabling or disabling general tablespace encryption

ALTER TABLESPACE ***tablespace\_name*** ENCRYPTION='Y';

[**ALTER TABLESPACE ... ENCRYPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) uses the **INPLACE** algorithm but does not support the **ALGORITHM** clause.

For related information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

Enabling or disabling file-per-table tablespace encryption

ALTER TABLE ***tbl\_name*** ENCRYPTION='Y', ALGORITHM=COPY;

For related information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

#### Partitioning Operations

With the exception of some [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) partitioning clauses, online DDL operations for partitioned **InnoDB** tables follow the same rules that apply to regular **InnoDB** tables.

Some [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) partitioning clauses do not go through the same internal online DDL API as regular non-partitioned **InnoDB** tables. As a result, online support for [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) partitioning clauses varies.

The following table shows the online status for each **ALTER TABLE** partitioning statement. Regardless of the online DDL API that is used, MySQL attempts to minimize data copying and locking where possible.

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) partitioning options that use **ALGORITHM=COPY** or that only permit “**ALGORITHM=DEFAULT, LOCK=DEFAULT**”, repartition the table using the **COPY** algorithm. In other words, a new partitioned table is created with the new partitioning scheme. The newly created table includes any changes applied by the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement, and table data is copied into the new table structure.

**Table 15.23 Online DDL Support for Partitioning Operations**

| **Partitioning Clause** | **Instant** | **In Place** | **Permits DML** | **Notes** |
| --- | --- | --- | --- | --- |
| [**PARTITION BY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | No | No | Permits **ALGORITHM=COPY**, **LOCK={DEFAULT|SHARED|EXCLUSIVE}** |
| [**ADD PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes\* | Yes\* | **ALGORITHM=INPLACE, LOCK={DEFAULT|NONE|SHARED|EXCLUSISVE}** is supported for **RANGE** and **LIST** partitions, **ALGORITHM=INPLACE, LOCK={DEFAULT|SHARED|EXCLUSISVE}** for **HASH** and **KEY** partitions, and **ALGORITHM=COPY, LOCK={SHARED|EXCLUSIVE}** for all partition types. Does not copy existing data for tables partitioned by **RANGE** or **LIST**. Concurrent queries are permitted with **ALGORITHM=COPY** for tables partitioned by **HASH** or **LIST**, as MySQL copies the data while holding a shared lock. |
| [**DROP PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes\* | Yes\* | **ALGORITHM=INPLACE, LOCK={DEFAULT|NONE|SHARED|EXCLUSIVE}** is supported. Does not copy data for tables partitioned by **RANGE** or **LIST**.  **DROP PARTITION** with **ALGORITHM=INPLACE** deletes data stored in the partition and drops the partition. However, **DROP PARTITION** with **ALGORITHM=COPY** or [**old\_alter\_table=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table) rebuilds the partitioned table and attempts to move data from the dropped partition to another partition with a compatible **PARTITION ... VALUES** definition. Data that cannot be moved to another partition is deleted. |
| [**DISCARD PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | No | No | Only permits **ALGORITHM=DEFAULT**, **LOCK=DEFAULT** |
| [**IMPORT PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | No | No | Only permits **ALGORITHM=DEFAULT**, **LOCK=DEFAULT** |
| [**TRUNCATE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes | Yes | Does not copy existing data. It merely deletes rows; it does not alter the definition of the table itself, or of any of its partitions. |
| [**COALESCE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes\* | No | **ALGORITHM=INPLACE, LOCK={DEFAULT|SHARED|EXCLUSIVE}** is supported. |
| [**REORGANIZE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes\* | No | **ALGORITHM=INPLACE, LOCK={DEFAULT|SHARED|EXCLUSIVE}** is supported. |
| [**EXCHANGE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes | Yes |  |
| [**ANALYZE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes | Yes |  |
| [**CHECK PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes | Yes |  |
| [**OPTIMIZE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | No | No | **ALGORITHM** and **LOCK** clauses are ignored. Rebuilds the entire table. See [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance). |
| [**REBUILD PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes\* | No | **ALGORITHM=INPLACE, LOCK={DEFAULT|SHARED|EXCLUSIVE}** is supported. |
| [**REPAIR PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | Yes | Yes |  |
| [**REMOVE PARTITIONING**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) | No | No | No | Permits **ALGORITHM=COPY**, **LOCK={DEFAULT|SHARED|EXCLUSIVE}** |

Non-partitioning online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations on partitioned tables follow the same rules that apply to regular tables. However, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) performs online operations on each table partition, which causes increased demand on system resources due to operations being performed on multiple partitions.

For additional information about [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) partitioning clauses, see [Partitioning Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-options), and [Section 13.1.9.1, “ALTER TABLE Partition Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations). For information about partitioning in general, see [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html).

### 15.12.2 Online DDL Performance and Concurrency

Online DDL improves several aspects of MySQL operation:

Applications that access the table are more responsive because queries and DML operations on the table can proceed while the DDL operation is in progress. Reduced locking and waiting for MySQL server resources leads to greater scalability, even for operations that are not involved in the DDL operation.

Instant operations only modify metadata in the data dictionary. No metadata locks are taken on the table, and table data is unaffected, making operations instantaneous. Concurrent DML is unaffected.

Online operations avoid the disk I/O and CPU cycles associated with the table-copy method, which minimizes overall load on the database. Minimizing load helps maintain good performance and high throughput during the DDL operation.

Online operations read less data into the buffer pool than table-copy operations, which reduces purging of frequently accessed data from memory. Purging of frequently accessed data can cause a temporary performance dip after a DDL operation.

#### The LOCK clause

By default, MySQL uses as little locking as possible during a DDL operation. The **LOCK** clause can be specified for in-place operations and some copy operations to enforce more restrictive locking, if required. If the **LOCK** clause specifies a less restrictive level of locking than is permitted for a particular DDL operation, the statement fails with an error. **LOCK** clauses are described below, in order of least to most restrictive:

**LOCK=NONE**:

Permits concurrent queries and DML.

For example, use this clause for tables involving customer signups or purchases, to avoid making the tables unavailable during lengthy DDL operations.

**LOCK=SHARED**:

Permits concurrent queries but blocks DML.

For example, use this clause on data warehouse tables, where you can delay data load operations until the DDL operation is finished, but queries cannot be delayed for long periods.

**LOCK=DEFAULT**:

Permits as much concurrency as possible (concurrent queries, DML, or both). Omitting the **LOCK** clause is the same as specifying **LOCK=DEFAULT**.

Use this clause when you do not expect the default locking level of the DDL statement to cause any availability problems for the table.

**LOCK=EXCLUSIVE**:

Blocks concurrent queries and DML.

Use this clause if the primary concern is finishing the DDL operation in the shortest amount of time possible, and concurrent query and DML access is not necessary. You might also use this clause if the server is supposed to be idle, to avoid unexpected table accesses.

#### Online DDL and Metadata Locks

Online DDL operations can be viewed as having three phases:

Phase 1: Initialization

In the initialization phase, the server determines how much concurrency is permitted during the operation, taking into account storage engine capabilities, operations specified in the statement, and user-specified **ALGORITHM** and **LOCK** options. During this phase, a shared upgradeable metadata lock is taken to protect the current table definition.

Phase 2: Execution

In this phase, the statement is prepared and executed. Whether the metadata lock is upgraded to exclusive depends on the factors assessed in the initialization phase. If an exclusive metadata lock is required, it is only taken briefly during statement preparation.

Phase 3: Commit Table Definition

In the commit table definition phase, the metadata lock is upgraded to exclusive to evict the old table definition and commit the new one. Once granted, the duration of the exclusive metadata lock is brief.

Due to the exclusive metadata lock requirements outlined above, an online DDL operation may have to wait for concurrent transactions that hold metadata locks on the table to commit or rollback. Transactions started before or during the DDL operation can hold metadata locks on the table being altered. In the case of a long running or inactive transaction, an online DDL operation can time out waiting for an exclusive metadata lock. Additionally, a pending exclusive metadata lock requested by an online DDL operation blocks subsequent transactions on the table.

The following example demonstrates an online DDL operation waiting for an exclusive metadata lock, and how a pending metadata lock blocks subsequent transactions on the table.

Session 1:

mysql> CREATE TABLE t1 (c1 INT) ENGINE=InnoDB;

mysql> START TRANSACTION;

mysql> SELECT \* FROM t1;

The session 1 [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement takes a shared metadata lock on table t1.

Session 2:

mysql> ALTER TABLE t1 ADD COLUMN x INT, ALGORITHM=INPLACE, LOCK=NONE;

The online DDL operation in session 2, which requires an exclusive metadata lock on table t1 to commit table definition changes, must wait for the session 1 transaction to commit or roll back.

Session 3:

mysql> SELECT \* FROM t1;

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement issued in session 3 is blocked waiting for the exclusive metadata lock requested by the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation in session 2 to be granted.

You can use [**SHOW FULL PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) to determine if transactions are waiting for a metadata lock.

mysql> **SHOW FULL PROCESSLIST\G**

...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 5

User: root

Host: localhost

db: test

Command: Query

Time: 44

State: Waiting for table metadata lock

Info: ALTER TABLE t1 ADD COLUMN x INT, ALGORITHM=INPLACE, LOCK=NONE

...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 7

User: root

Host: localhost

db: test

Command: Query

Time: 5

State: Waiting for table metadata lock

Info: SELECT \* FROM t1

4 rows in set (0.00 sec)

Metadata lock information is also exposed through the Performance Schema [**metadata\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-metadata-locks-table) table, which provides information about metadata lock dependencies between sessions, the metadata lock a session is waiting for, and the session that currently holds the metadata lock. For more information, see [Section 27.12.13.3, “The metadata\_locks Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-metadata-locks-table).

#### Online DDL Performance

The performance of a DDL operation is largely determined by whether the operation is performed instantly, in place, and whether it rebuilds the table.

To assess the relative performance of a DDL operation, you can compare results using **ALGORITHM=INSTANT**, **ALGORITHM=INPLACE**, and **ALGORITHM=COPY**. A statement can also be run with [**old\_alter\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table) enabled to force the use of **ALGORITHM=COPY**.

For DDL operations that modify table data, you can determine whether a DDL operation performs changes in place or performs a table copy by looking at the “rows affected” value displayed after the command finishes. For example:

Changing the default value of a column (fast, does not affect the table data):

Query OK, 0 rows affected (0.07 sec)

Adding an index (takes time, but **0 rows affected** shows that the table is not copied):

Query OK, 0 rows affected (21.42 sec)

Changing the data type of a column (takes substantial time and requires rebuilding all the rows of the table):

Query OK, 1671168 rows affected (1 min 35.54 sec)

Before running a DDL operation on a large table, check whether the operation is fast or slow as follows:

Clone the table structure.

Populate the cloned table with a small amount of data.

Run the DDL operation on the cloned table.

Check whether the “rows affected” value is zero or not. A nonzero value means the operation copies table data, which might require special planning. For example, you might do the DDL operation during a period of scheduled downtime, or on each replica server one at a time.

**Note**

For a greater understanding of the MySQL processing associated with a DDL operation, examine Performance Schema and **INFORMATION\_SCHEMA** tables related to **InnoDB** before and after DDL operations to see the number of physical reads, writes, memory allocations, and so on.

Performance Schema stage events can be used to monitor [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) progress. See [Section 15.16.1, “Monitoring ALTER TABLE Progress for InnoDB Tables Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-alter-table-performance-schema).

Because there is some processing work involved with recording the changes made by concurrent DML operations, then applying those changes at the end, an online DDL operation could take longer overall than the table-copy mechanism that blocks table access from other sessions. The reduction in raw performance is balanced against better responsiveness for applications that use the table. When evaluating the techniques for changing table structure, consider end-user perception of performance, based on factors such as load times for web pages.

### 15.12.3 Online DDL Space Requirements

Space requirements for in-place online DDL operations are outlined below. Space requirements do not apply to operations that are performed instantly.

Space for temporary log files

A temporary log file records concurrent DML when an online DDL operation creates an index or alters a table. The temporary log file is extended as required by the value of [**innodb\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sort_buffer_size) up to a maximum specified by [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size). If a temporary log file exceeds the size limit, the online DDL operation fails, and uncommitted concurrent DML operations are rolled back. A large [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size) setting permits more DML during an online DDL operation, but it also extends the period of time at the end of the DDL operation when the table is locked to apply logged DML.

If the operation takes a long time and concurrent DML modifies the table so much that the size of the temporary log file exceeds the value of [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size), the online DDL operation fails with a **DB\_ONLINE\_LOG\_TOO\_BIG** error.

Space for temporary sort files

Online DDL operations that rebuild the table write temporary sort files to the MySQL temporary directory (**$TMPDIR** on Unix, **%TEMP%** on Windows, or the directory specified by [**--tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir)) during index creation. Temporary sort files are not created in the directory that contains the original table. Each temporary sort file is large enough to hold one column of data, and each sort file is removed when its data is merged into the final table or index. Operations involving temporary sort files may require temporary space equal to the amount of data in the table plus indexes. An error is reported if online DDL operation uses all of the available disk space on the file system where the data directory resides.

If the MySQL temporary directory is not large enough to hold the sort files, set [**tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) to a different directory. Alternatively, define a separate temporary directory for online DDL operations using [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir). This option was introduced to help avoid temporary directory overflows that could occur as a result of large temporary sort files.

Space for an intermediate table file

Some online DDL operations that rebuild the table create a temporary intermediate table file in the same directory as the original table. An intermediate table file may require space equal to the size of the original table. Intermediate table file names begin with #sql-ib prefix and only appear briefly during the online DDL operation.

The [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) option is not applicable to intermediate table files.

### 15.12.4 Simplifying DDL Statements with Online DDL

Before the introduction of [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_online_ddl), it was common practice to combine many DDL operations into a single [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. Because each [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement involved copying and rebuilding the table, it was more efficient to make several changes to the same table at once, since those changes could all be done with a single rebuild operation for the table. The downside was that SQL code involving DDL operations was harder to maintain and to reuse in different scripts. If the specific changes were different each time, you might have to construct a new complex [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) for each slightly different scenario.

For DDL operations that can be done online, you can separate them into individual [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements for easier scripting and maintenance, without sacrificing efficiency. For example, you might take a complicated statement such as:

ALTER TABLE t1 ADD INDEX i1(c1), ADD UNIQUE INDEX i2(c2),

CHANGE c4\_old\_name c4\_new\_name INTEGER UNSIGNED;

and break it down into simpler parts that can be tested and performed independently, such as:

ALTER TABLE t1 ADD INDEX i1(c1);

ALTER TABLE t1 ADD UNIQUE INDEX i2(c2);

ALTER TABLE t1 CHANGE c4\_old\_name c4\_new\_name INTEGER UNSIGNED NOT NULL;

You might still use multi-part [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements for:

Operations that must be performed in a specific sequence, such as creating an index followed by a foreign key constraint that uses that index.

Operations all using the same specific **LOCK** clause, that you want to either succeed or fail as a group.

Operations that cannot be performed online, that is, that still use the table-copy method.

Operations for which you specify **ALGORITHM=COPY** or [**old\_alter\_table=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table), to force the table-copying behavior if needed for precise backward-compatibility in specialized scenarios.

### 15.12.5 Online DDL Failure Conditions

The failure of an online DDL operation is typically due to one of the following conditions:

An **ALGORITHM** clause specifies an algorithm that is not compatible with the particular type of DDL operation or storage engine.

A **LOCK** clause specifies a low degree of locking (**SHARED** or **NONE**) that is not compatible with the particular type of DDL operation.

A timeout occurs while waiting for an [exclusive lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_exclusive_lock) on the table, which may be needed briefly during the initial and final phases of the DDL operation.

The [**tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) or [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) file system runs out of disk space, while MySQL writes temporary sort files on disk during index creation. For more information, see [Section 15.12.3, “Online DDL Space Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-space-requirements).

The operation takes a long time and concurrent DML modifies the table so much that the size of the temporary online log exceeds the value of the [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size) configuration option. This condition causes a **DB\_ONLINE\_LOG\_TOO\_BIG** error.

Concurrent DML makes changes to the table that are allowed with the original table definition, but not with the new one. The operation only fails at the very end, when MySQL tries to apply all the changes from concurrent DML statements. For example, you might insert duplicate values into a column while a unique index is being created, or you might insert **NULL** values into a column while creating a [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) index on that column. The changes made by the concurrent DML take precedence, and the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation is effectively [rolled back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback).

### 15.12.6 Online DDL Limitations

The following limitations apply to online DDL operations:

The table is copied when creating an index on a **TEMPORARY TABLE**.

The [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) clause **LOCK=NONE** is not permitted if there are **ON...CASCADE** or **ON...SET NULL** constraints on the table.

Before an in-place online DDL operation can finish, it must wait for transactions that hold metadata locks on the table to commit or roll back. An online DDL operation may briefly require an exclusive metadata lock on the table during its execution phase, and always requires one in the final phase of the operation when updating the table definition. Consequently, transactions holding metadata locks on the table can cause an online DDL operation to block. The transactions that hold metadata locks on the table may have been started before or during the online DDL operation. A long running or inactive transaction that holds a metadata lock on the table can cause an online DDL operation to timeout.

When running an in-place online DDL operation, the thread that runs the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement applies an online log of DML operations that were run concurrently on the same table from other connection threads. When the DML operations are applied, it is possible to encounter a duplicate key entry error (ERROR 1062 (23000): Duplicate entry), even if the duplicate entry is only temporary and would be reverted by a later entry in the online log. This is similar to the idea of a foreign key constraint check in **InnoDB** in which constraints must hold during a transaction.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) for an **InnoDB** table is mapped to an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation to rebuild the table and update index statistics and free unused space in the clustered index. Secondary indexes are not created as efficiently because keys are inserted in the order they appeared in the primary key. [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is supported with the addition of online DDL support for rebuilding regular and partitioned **InnoDB** tables.

Tables created before MySQL 5.6 that include temporal columns ([**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime)) and have not been rebuilt using **ALGORITHM=COPY** do not support **ALGORITHM=INPLACE**. In this case, an [**ALTER TABLE ... ALGORITHM=INPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation returns the following error:

ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported.

Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY.

The following limitations are generally applicable to online DDL operations on large tables that involve rebuilding the table:

There is no mechanism to pause an online DDL operation or to throttle I/O or CPU usage for an online DDL operation.

Rollback of an online DDL operation can be expensive should the operation fail.

Long running online DDL operations can cause replication lag. An online DDL operation must finish running on the source before it is run on the replica. Also, DML that was processed concurrently on the source is only processed on the replica after the DDL operation on the replica is completed.

For additional information related to running online DDL operations on large tables, see [Section 15.12.2, “Online DDL Performance and Concurrency”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-performance).

## 15.13 InnoDB Data-at-Rest Encryption

**InnoDB** supports data-at-rest encryption for [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces, [general](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) tablespaces, the **mysql** system tablespace, redo logs, and undo logs.

As of MySQL 8.0.16, setting an encryption default for schemas and general tablespaces is also supported, which permits DBAs to control whether tables created in those schemas and tablespaces are encrypted.

**InnoDB** data-at-rest encryption features and capabilities are described under the following topics in this section.

[About Data-at-Rest Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-about)

[Encryption Prerequisites](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-encryption-prerequisites)

[Defining an Encryption Default for Schemas and General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-schema-tablespace-encryption-default)

[File-Per-Table Tablespace Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-enabling-disabling)

[General Tablespace Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-general-tablespace-encryption-enabling-disabling)

[Doublewrite File Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-file-encryption)

[mysql System Tablespace Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-mysql-tablespace-encryption-enabling-disabling)

[Redo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-redo-log)

[Undo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-undo-log)

[Master Key Rotation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-master-key-rotation)

[Encryption and Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-recovery)

[Exporting Encrypted Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-exporting)

[Encryption and Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-replication)

[Identifying Encrypted Tablespaces and Schemas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-identifying)

[Monitoring Encryption Progress](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-progress-monitoring)

[Encryption Usage Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-usage-notes)

[Encryption Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-limitations)

### About Data-at-Rest Encryption

**InnoDB** uses a two tier encryption key architecture, consisting of a master encryption key and tablespace keys. When a tablespace is encrypted, a tablespace key is encrypted and stored in the tablespace header. When an application or authenticated user wants to access encrypted tablespace data, **InnoDB** uses a master encryption key to decrypt the tablespace key. The decrypted version of a tablespace key never changes, but the master encryption key can be changed as required. This action is referred to as master key rotation.

The data-at-rest encryption feature relies on a keyring component or plugin for master encryption key management.

All MySQL editions provide a **component\_keyring\_file** component and **keyring\_file** plugin, each of which stores keyring data in a file local to the server host.

MySQL Enterprise Edition offers additional keyring components and plugins:

**component\_keyring\_encrypted\_file**: Stores keyring data in an encrypted, password-protected file local to the server host.

**keyring\_encrypted\_file**: Stores keyring data in an encrypted, password-protected file local to the server host.

**keyring\_okv**: A KMIP 1.1 plugin for use with KMIP-compatible back end keyring storage products. Supported KMIP-compatible products include centralized key management solutions such as Oracle Key Vault, Gemalto KeySecure, Thales Vormetric key management server, and Fornetix Key Orchestration.

**keyring\_aws**: Communicates with the Amazon Web Services Key Management Service (AWS KMS) as a back end for key generation and uses a local file for key storage.

**keyring\_hashicorp**: Communicates with HashiCorp Vault for back end storage.

**Warning**

For encryption key management, the **component\_keyring\_file** and **component\_keyring\_encrypted\_file** components, and the **keyring\_file** and **keyring\_encrypted\_file** plugins are not intended as a regulatory compliance solution. Security standards such as PCI, FIPS, and others require use of key management systems to secure, manage, and protect encryption keys in key vaults or hardware security modules (HSMs).

A secure and robust encryption key management solution is critical for security and for compliance with various security standards. When the data-at-rest encryption feature uses a centralized key management solution, the feature is referred to as “MySQL Enterprise Transparent Data Encryption (TDE)”.

The data-at-rest encryption feature supports the Advanced Encryption Standard (AES) block-based encryption algorithm. It uses Electronic Codebook (ECB) block encryption mode for tablespace key encryption and Cipher Block Chaining (CBC) block encryption mode for data encryption.

For frequently asked questions about the data-at-rest encryption feature, see [Section A.17, “MySQL 8.0 FAQ: InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\faqs.html#faqs-tablespace-encryption).

### Encryption Prerequisites

A keyring component or plugin must be installed and configured at startup. Early loading ensures that the component or plugin is available prior to initialization of the **InnoDB** storage engine. For keyring installation and configuration instructions, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring). The instructions show how to ensure that the chosen component or plugin is active.

Only one keyring component or plugin should be enabled at a time. Enabling multiple keyring components or plugins is unsupported and results may not be as anticipated.

**Important**

Once encrypted tablespaces are created in a MySQL instance, the keyring component or plugin that was loaded when creating the encrypted tablespace must continue to be loaded at startup. Failing to do so results in errors when starting the server and during **InnoDB** recovery.

When encrypting production data, ensure that you take steps to prevent loss of the master encryption key. If the master encryption key is lost, data stored in encrypted tablespace files is unrecoverable. If you use the **component\_keyring\_file** or **component\_keyring\_encrypted\_file** component, or the **keyring\_file** or **keyring\_encrypted\_file** plugin, create a backup of the keyring data file immediately after creating the first encrypted tablespace, before master key rotation, and after master key rotation. For each component, its configuration file indicates the data file location. The [**keyring\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_file_data) configuration option defines the keyring data file location for the **keyring\_file** plugin. The [**keyring\_encrypted\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_encrypted_file_data) configuration option defines the keyring data file location for the **keyring\_encrypted\_file** plugin. If you use the **keyring\_okv** or **keyring\_aws** plugin, ensure that you have performed the necessary configuration. For instructions, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring).

### Defining an Encryption Default for Schemas and General Tablespaces

As of MySQL 8.0.16, the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) system variable defines the default encryption setting for schemas and general tablespaces. [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) and [**CREATE SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) operations apply the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting when an **ENCRYPTION** clause is not specified explicitly.

[**ALTER SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) and [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) operations do not apply the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting. An **ENCRYPTION** clause must be specified explicitly to alter the encryption of an existing schema or general tablespace.

The [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) variable can be set for an individual client connection or globally using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) syntax. For example, the following statement enables default schema and tablespace encryption globally:

mysql> SET GLOBAL default\_table\_encryption=ON;

The default encryption setting for a schema can also be defined using the **DEFAULT ENCRYPTION** clause when creating or altering a schema, as in this example:

mysql> CREATE SCHEMA test DEFAULT ENCRYPTION = 'Y';

If the **DEFAULT ENCRYPTION** clause is not specified when creating a schema, the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting is applied. The **DEFAULT ENCRYPTION** clause must be specified to alter the default encryption of an existing schema. Otherwise, the schema retains its current encryption setting.

By default, a table inherits the encryption setting of the schema or general tablespace it is created in. For example, a table created in an encryption-enabled schema is encrypted by default. This behavior enables a DBA to control table encryption usage by defining and enforcing schema and general tablespace encryption defaults.

Encryption defaults are enforced by enabling the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) system variable. When [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) is enabled, a privilege check occurs when creating or altering a schema or general tablespace with an encryption setting that differs from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting, or when creating or altering a table with an encryption setting that differs from the default schema encryption. When [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) is disabled (the default), the privilege check does not occur and the previously mentioned operations are permitted to proceed with a warning.

The [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to override default encryption settings when [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) is enabled. A DBA can grant this privilege to enable a user to deviate from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting when creating or altering a schema or general tablespace, or to deviate from the default schema encryption when creating or altering a table. This privilege does not permit deviating from the encryption of a general tablespace when creating or altering a table. A table must have the same encryption setting as the general tablespace it resides in.

### File-Per-Table Tablespace Encryption

As of MySQL 8.0.16, a file-per-table tablespace inherits the default encryption of the schema in which the table is created unless an **ENCRYPTION** clause is specified explcitly in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. Prior to MySQL 8.0.16, the **ENCRYPTION** clause must be specified to enable encryption.

mysql> **CREATE TABLE t1 (c1 INT) ENCRYPTION = 'Y';**

To alter the encryption of an existing file-per-table tablespace, an **ENCRYPTION** clause must be specified.

mysql> **ALTER TABLE t1 ENCRYPTION = 'Y';**

As of MySQL 8.0.16, if the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, specifying an **ENCRYPTION** clause with a setting that differs from the default schema encryption requires the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege. See [Defining an Encryption Default for Schemas and General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-schema-tablespace-encryption-default).

### General Tablespace Encryption

As of MySQL 8.0.16, the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) variable determines the encryption of a newly created general tablespace unless an **ENCRYPTION** clause is specified explicitly in the [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) statement. Prior to MySQL 8.0.16, an **ENCRYPTION** clause must be specified to enable encryption.

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' ENCRYPTION = 'Y' Engine=InnoDB;**

To alter the encryption of an existing general tablespace, an **ENCRYPTION** clause must be specified.

mysql> **ALTER TABLESPACE ts1 ENCRYPTION = 'Y';**

As of MySQL 8.0.16, if the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, specifying an **ENCRYPTION** clause with a setting that differs from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting requires the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege. See [Defining an Encryption Default for Schemas and General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-schema-tablespace-encryption-default).

### Doublewrite File Encryption

Encryption support for doublewrite files is available as of MySQL 8.0.23. **InnoDB** automatically encrypts doublewrite file pages that belong to encrypted tablespaces. No action is required. Doublewrite file pages are encrypted using the encryption key of the associated tablespace. The same encrypted page written to a tablespace data file is also written to a doublewrite file. Doublewrite file pages that belong to an unencrypted tablespace remain unencrypted.

During recovery, encrypted doublewrite file pages are unencrypted and checked for corruption.

### mysql System Tablespace Encryption

Encryption support for the **mysql** system tablespace is available as of MySQL 8.0.16.

The **mysql** system tablespace contains the **mysql** system database and MySQL data dictionary tables. It is unencrypted by default. To enable encryption for the **mysql** system tablespace, specify the tablespace name and the **ENCRYPTION** option in an [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement.

mysql> ALTER TABLESPACE mysql ENCRYPTION = 'Y';

To disable encryption for the **mysql** system tablespace, set **ENCRYPTION = 'N'** using an [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement.

mysql> ALTER TABLESPACE mysql ENCRYPTION = 'N';

Enabling or disabling encryption for the **mysql** system tablespace requires the [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-tablespace) privilege on all tables in the instance (**CREATE TABLESPACE on \*.\*)**.

### Redo Log Encryption

Redo log data encryption is enabled using the [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) configuration option. Redo log encryption is disabled by default.

As with tablespace data, redo log data encryption occurs when redo log data is written to disk, and decryption occurs when redo log data is read from disk. Once redo log data is read into memory, it is in unencrypted form. Redo log data is encrypted and decrypted using the tablespace encryption key.

When [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) is enabled, unencrypted redo log pages that are present on disk remain unencrypted, and new redo log pages are written to disk in encrypted form. Likewise, when [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) is disabled, encrypted redo log pages that are present on disk remain encrypted, and new redo log pages are written to disk in unencrypted form.

Redo log encryption metadata, including the tablespace encryption key, is stored in the header of the first redo log file (ib\_logfile0). If this file is removed, redo log encryption is disabled.

Once redo log encryption is enabled, a normal restart without the keyring component or plugin or without the encryption key is not possible, as **InnoDB** must be able to scan redo pages during startup, which is not possible if redo log pages are encrypted. Without the keyring component or plugin or the encryption key, only a forced startup without the redo logs (**SRV\_FORCE\_NO\_LOG\_REDO**) is possible. See [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

### Undo Log Encryption

Undo log data encryption is enabled using the [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt) configuration option. Undo log encryption applies to undo logs that reside in [undo tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace). See [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces). Undo log data encryption is disabled by default.

As with tablespace data, undo log data encryption occurs when undo log data is written to disk, and decryption occurs when undo log data is read from disk. Once undo log data is read into memory, it is in unencrypted form. Undo log data is encrypted and decrypted using the tablespace encryption key.

When [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt) is enabled, unencrypted undo log pages that are present on disk remain unencrypted, and new undo log pages are written to disk in encrypted form. Likewise, when [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt) is disabled, encrypted undo log pages that are present on disk remain encrypted, and new undo log pages are written to disk in unencrypted form.

Undo log encryption metadata, including the tablespace encryption key, is stored in the header of the undo log file.

**Note**

When undo log encryption is disabled, the server continues to require the keyring component or plugin that was used to encrypt undo log data until the undo tablespaces that contained the encrypted undo log data are truncated. (An encryption header is only removed from an undo tablespace when the undo tablespace is truncated.) For information about truncating undo tablespaces, see [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

### Master Key Rotation

The master encryption key should be rotated periodically and whenever you suspect that the key has been compromised.

Master key rotation is an atomic, instance-level operation. Each time the master encryption key is rotated, all tablespace keys in the MySQL instance are re-encrypted and saved back to their respective tablespace headers. As an atomic operation, re-encryption must succeed for all tablespace keys once a rotation operation is initiated. If master key rotation is interrupted by a server failure, **InnoDB** rolls the operation forward on server restart. For more information, see [Encryption and Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-recovery).

Rotating the master encryption key only changes the master encryption key and re-encrypts tablespace keys. It does not decrypt or re-encrypt associated tablespace data.

Rotating the master encryption key requires the [**ENCRYPTION\_KEY\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_encryption-key-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

To rotate the master encryption key, run:

mysql> **ALTER INSTANCE ROTATE INNODB MASTER KEY;**

[**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) supports concurrent DML. However, it cannot be run concurrently with tablespace encryption operations, and locks are taken to prevent conflicts that could arise from concurrent execution. If an [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) operation is running, it must finish before a tablespace encryption operation can proceed, and vice versa.

### Encryption and Recovery

If a server failure occurs during an encryption operation, the operation is rolled forward when the server is restarted. For general tablespaces, the encryption operation is resumed in a background thread from the last processed page.

If a server failure occurs during master key rotation, **InnoDB** continues the operation on server restart.

The keyring component or plugin must be loaded prior to storage engine initialization so that the information necessary to decrypt tablespace data pages can be retrieved from tablespace headers before **InnoDB** initialization and recovery activities access tablespace data. (See [Encryption Prerequisites](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-encryption-prerequisites).)

When **InnoDB** initialization and recovery begin, the master key rotation operation resumes. Due to the server failure, some tablespace keys may already be encrypted using the new master encryption key. **InnoDB** reads the encryption data from each tablespace header, and if the data indicates that the tablespace key is encrypted using the old master encryption key, **InnoDB** retrieves the old key from the keyring and uses it to decrypt the tablespace key. **InnoDB** then re-encrypts the tablespace key using the new master encryption key and saves the re-encrypted tablespace key back to the tablespace header.

### Exporting Encrypted Tablespaces

Tablespace export is only supported for file-per-table tablespaces.

When an encrypted tablespace is exported, **InnoDB** generates a transfer key that is used to encrypt the tablespace key. The encrypted tablespace key and transfer key are stored in a ***tablespace\_name***.cfp file. This file together with the encrypted tablespace file is required to perform an import operation. On import, **InnoDB** uses the transfer key to decrypt the tablespace key in the ***tablespace\_name***.cfp file. For related information, see [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

### Encryption and Replication

The [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) statement is only supported in replication environments where the source and replica run a version of MySQL that supports tablespace encryption.

Successful [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) statements are written to the binary log for replication on replicas.

If an [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) statement fails, it is not logged to the binary log and is not replicated on replicas.

Replication of an [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) operation fails if the keyring component or plugin is installed on the source but not on the replica.

If the **keyring\_file** or **keyring\_encrypted\_file** plugin is installed on both the source and a replica but the replica does not have a keyring data file, the replicated [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) statement creates the keyring data file on the replica, assuming the keyring file data is not cached in memory. [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) uses keyring file data that is cached in memory, if available.

### Identifying Encrypted Tablespaces and Schemas

The [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table, introduced in MySQL 8.0.13, includes an **ENCRYPTION** column that can be used to identify encrypted tablespaces.

mysql> **SELECT SPACE, NAME, SPACE\_TYPE, ENCRYPTION FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES**

**WHERE ENCRYPTION='Y'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 4294967294

NAME: mysql

SPACE\_TYPE: General

ENCRYPTION: Y

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 2

NAME: test/t1

SPACE\_TYPE: Single

ENCRYPTION: Y

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 3

NAME: ts1

SPACE\_TYPE: General

ENCRYPTION: Y

When the **ENCRYPTION** option is specified in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement, it is recorded in the **CREATE\_OPTIONS** column of [**INFORMATION\_SCHEMA.TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table). This column can be queried to identify tables that reside in encrypted file-per-table tablespaces.

mysql> **SELECT TABLE\_SCHEMA, TABLE\_NAME, CREATE\_OPTIONS FROM INFORMATION\_SCHEMA.TABLES**

**WHERE CREATE\_OPTIONS LIKE '%ENCRYPTION%';**

+--------------+------------+----------------+

| TABLE\_SCHEMA | TABLE\_NAME | CREATE\_OPTIONS |

+--------------+------------+----------------+

| test | t1 | ENCRYPTION="Y" |

+--------------+------------+----------------+

Query [**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) to retrieve information about the tablespace associated with a particular schema and table.

mysql> **SELECT SPACE, NAME, SPACE\_TYPE FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES WHERE NAME='test/t1';**

+-------+---------+------------+

| SPACE | NAME | SPACE\_TYPE |

+-------+---------+------------+

| 3 | test/t1 | Single |

+-------+---------+------------+

You can identify encryption-enabled schemas by querying the [**INFORMATION\_SCHEMA.SCHEMATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table) table.

mysql> **SELECT SCHEMA\_NAME, DEFAULT\_ENCRYPTION FROM INFORMATION\_SCHEMA.SCHEMATA**

**WHERE DEFAULT\_ENCRYPTION='YES';**

+-------------+--------------------+

| SCHEMA\_NAME | DEFAULT\_ENCRYPTION |

+-------------+--------------------+

| test | YES |

+-------------+--------------------+

[**SHOW CREATE SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database) also shows the **DEFAULT ENCRYPTION** clause.

### Monitoring Encryption Progress

You can monitor general tablespace and **mysql** system tablespace encryption progress using [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html).

The **stage/innodb/alter tablespace (encryption)** stage event instrument reports **WORK\_ESTIMATED** and **WORK\_COMPLETED** information for general tablespace encryption operations.

The following example demonstrates how to enable the **stage/innodb/alter tablespace (encryption)** stage event instrument and related consumer tables to monitor general tablespace or **mysql** system tablespace encryption progress. For information about Performance Schema stage event instruments and related consumers, see [Section 27.12.5, “Performance Schema Stage Event Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-tables).

Enable the **stage/innodb/alter tablespace (encryption)** instrument:

mysql> **USE performance\_schema;**

mysql> **UPDATE setup\_instruments SET ENABLED = 'YES'**

**WHERE NAME LIKE 'stage/innodb/alter tablespace (encryption)';**

Enable the stage event consumer tables, which include [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table), [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table), and [**events\_stages\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-long-table).

mysql> **UPDATE setup\_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%stages%';**

Run a tablespace encryption operation. In this example, a general tablespace named **ts1** is encrypted.

mysql> **ALTER TABLESPACE ts1 ENCRYPTION = 'Y';**

Check the progress of the encryption operation by querying the Performance Schema [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table. **WORK\_ESTIMATED** reports the total number of pages in the tablespace. **WORK\_COMPLETED** reports the number of pages processed.

mysql> **SELECT EVENT\_NAME, WORK\_ESTIMATED, WORK\_COMPLETED FROM events\_stages\_current;**

+--------------------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+--------------------------------------------+----------------+----------------+

| stage/innodb/alter tablespace (encryption) | 1056 | 1407 |

+--------------------------------------------+----------------+----------------+

The [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table returns an empty set if the encryption operation has completed. In this case, you can check the [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table) table to view event data for the completed operation. For example:

mysql> **SELECT EVENT\_NAME, WORK\_COMPLETED, WORK\_ESTIMATED FROM events\_stages\_history;**

+--------------------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+--------------------------------------------+----------------+----------------+

| stage/innodb/alter tablespace (encryption) | 1407 | 1407 |

+--------------------------------------------+----------------+----------------+

### Encryption Usage Notes

Plan appropriately when altering an existing file-per-table tablespace with the **ENCRYPTION** option. Tables residing in file-per-table tablespaces are rebuilt using the **COPY** algorithm. The **INPLACE** algorithm is used when altering the **ENCRYPTION** attribute of a general tablespace or the **mysql** system tablespace. The **INPLACE** algorithm permits concurrent DML on tables that reside in the general tablespace. Concurrent DDL is blocked.

When a general tablespace or the **mysql** system tablespace is encrypted, all tables residing in the tablespace are encrypted. Likewise, a table created in an encrypted tablespace is encrypted.

If the server exits or is stopped during normal operation, it is recommended to restart the server using the same encryption settings that were configured previously.

The first master encryption key is generated when the first new or existing tablespace is encrypted.

Master key rotation re-encrypts tablespaces keys but does not change the tablespace key itself. To change a tablespace key, you must disable and re-enable encryption. For file-per-table tablespaces, re-encrypting the tablespace is an **ALGORITHM=COPY** operation that rebuilds the table. For general tablespaces and the **mysql** system tablespace, it is an **ALGORITHM=INPLACE** operation, which does not require rebuilding tables that reside in the tablespace.

If a table is created with both the [**COMPRESSION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ENCRYPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) options, compression is performed before tablespace data is encrypted.

If a keyring data file (the file named by [**keyring\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_file_data) or [**keyring\_encrypted\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_encrypted_file_data)) is empty or missing, the first execution of [**ALTER INSTANCE ROTATE INNODB MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-innodb-master-key) creates a master encryption key.

Uninstalling the **component\_keyring\_file** or **component\_keyring\_encrypted\_file** component does not remove an existing keyring data file. Uninstalling the **keyring\_file** or **keyring\_encrypted\_file** plugin does not remove an existing keyring data file.

It is recommended that you not place a keyring data file under the same directory as tablespace data files.

Modifying the [**keyring\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_file_data) or [**keyring\_encrypted\_file\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#sysvar_keyring_encrypted_file_data) setting at runtime or when restarting the server can cause previously encrypted tablespaces to become inaccessible, resulting in lost data.

Encryption is supported for the **InnoDB** **FULLTEXT** index tables that are created implicitly when adding a **FULLTEXT** index, but only if the **FULLTEXT** index is created on a table that resides in an encrypted general tablespace. In this case, the **FULLTEXT** index tables are created in the same encrypted general tablespace. For related information, see [InnoDB Full-Text Index Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-tables).

### Encryption Limitations

Advanced Encryption Standard (AES) is the only supported encryption algorithm. **InnoDB** tablespace encryption uses Electronic Codebook (ECB) block encryption mode for tablespace key encryption and Cipher Block Chaining (CBC) block encryption mode for data encryption. Padding is not used with CBC block encryption mode. Instead, **InnoDB** ensures that the text to be encrypted is a multiple of the block size.

Encryption is only supported for [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces, [general](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) tablespaces, and the **mysql** system tablespace. Encryption support for general tablespaces was introduced in MySQL 8.0.13. Encryption support for the **mysql** system tablespace is available as of MySQL 8.0.16. Encryption is not supported for other tablespace types including the **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace).

You cannot move or copy a table from an encrypted [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace, [general](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) tablespace, or the **mysql** system tablespace to a tablespace type that does not support encryption.

You cannot move or copy a table from an encrypted tablespace to an unencrypted tablespace. However, moving a table from an unencrypted tablespace to an encrypted one is permitted. For example, you can move or copy a table from a unencrypted [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) or [general](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) tablespace to an encrypted general tablespace.

By default, tablespace encryption only applies to data in the tablespace. Redo log and undo log data can be encrypted by enabling [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) and [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt). See [Redo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-redo-log), and [Undo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-undo-log). For information about binary log file and relay log file encryption, see [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption).

It is not permitted to change the storage engine of a table that resides in, or previously resided in, an encrypted tablespace.

## 15.14 InnoDB Startup Options and System Variables

System variables that are true or false can be enabled at server startup by naming them, or disabled by using a **--skip-** prefix. For example, to enable or disable the **InnoDB** adaptive hash index, you can use [--innodb-adaptive-hash-index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) or [--skip-innodb-adaptive-hash-index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) on the command line, or [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) or **skip\_innodb\_adaptive\_hash\_index** in an option file.

System variables that take a numeric value can be specified as --***var\_name***=***value*** on the command line or as ***var\_name*=*value*** in option files.

Many system variables can be changed at runtime (see [Section 5.1.9.2, “Dynamic System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#dynamic-system-variables)).

For information about **GLOBAL** and **SESSION** variable scope modifiers, refer to the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement documentation.

Certain options control the locations and layout of the **InnoDB** data files. [Section 15.8.1, “InnoDB Startup Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-init-startup-configuration) explains how to use these options.

Some options, which you might not use initially, help tune **InnoDB** performance characteristics based on machine capacity and your database [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload).

For more information on specifying options and system variables, see [Section 4.2.2, “Specifying Program Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#program-options).

**Table 15.24 InnoDB Option and Variable Reference**

| **Name** | **Cmd-Line** | **Option File** | **System Var** | **Status Var** | **Var Scope** | **Dynamic** |
| --- | --- | --- | --- | --- | --- | --- |
| [**daemon\_memcached\_enable\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_enable_binlog) | Yes | Yes | Yes |  | Global | No |
| [**daemon\_memcached\_engine\_lib\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_name) | Yes | Yes | Yes |  | Global | No |
| [**daemon\_memcached\_engine\_lib\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_path) | Yes | Yes | Yes |  | Global | No |
| [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) | Yes | Yes | Yes |  | Global | No |
| [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) | Yes | Yes | Yes |  | Global | No |
| [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) | Yes | Yes | Yes |  | Global | No |
| [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) |  |  | Yes |  | Both | Yes |
| [**innodb**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb) | Yes | Yes |  |  |  |  |
| [**innodb\_adaptive\_flushing**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_adaptive\_flushing\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing_lwm) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_adaptive\_hash\_index\_parts**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index_parts) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_adaptive\_max\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_max_sleep_delay) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_api\_bk\_commit\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_bk_commit_interval) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_api\_enable\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_binlog) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_api\_enable\_mdl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_mdl) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_background\_drop\_list\_empty**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_background_drop_list_empty) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_buffer\_pool\_bytes\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_bytes_data) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_bytes\_dirty**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_bytes_dirty) |  |  |  | Yes | Global | No |
| [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_buffer\_pool\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_debug) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_buffer\_pool\_dump\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_now) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_buffer\_pool\_dump\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_dump_status) |  |  |  | Yes | Global | No |
| [**innodb\_buffer\_pool\_filename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_filename) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_buffer\_pool\_load\_abort**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_abort) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_buffer\_pool\_load\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_now) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_buffer\_pool\_load\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_load_status) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_data) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_dirty**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_dirty) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_flushed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_flushed) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_free**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_free) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_latched**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_latched) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_misc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_misc) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_pages\_total**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_pages_total) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_read\_ahead\_evicted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead_evicted) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_read\_ahead\_rnd**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead_rnd) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_read\_requests**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_requests) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_reads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_reads) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_resize\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_resize_status) |  |  |  | Yes | Global | No |
| [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_buffer\_pool\_wait\_free**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_wait_free) |  |  |  | Yes | Global | No |
| [**Innodb\_buffer\_pool\_write\_requests**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_write_requests) |  |  |  | Yes | Global | No |
| [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_change\_buffering\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_checkpoint\_disabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checkpoint_disabled) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_checksum\_algorithm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checksum_algorithm) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_cmp\_per\_index\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_cmp_per_index_enabled) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_commit\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_commit_concurrency) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_compress\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compress_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_level) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) | Yes | Yes | Yes |  | Global | No |
| [**Innodb\_data\_fsyncs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_fsyncs) |  |  |  | Yes | Global | No |
| [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) | Yes | Yes | Yes |  | Global | No |
| [**Innodb\_data\_pending\_fsyncs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_pending_fsyncs) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_pending\_reads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_pending_reads) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_pending\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_pending_writes) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_read**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_read) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_reads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_reads) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_writes) |  |  |  | Yes | Global | No |
| [**Innodb\_data\_written**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_written) |  |  |  | Yes | Global | No |
| [**Innodb\_dblwr\_pages\_written**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_dblwr_pages_written) |  |  |  | Yes | Global | No |
| [**Innodb\_dblwr\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_dblwr_writes) |  |  |  | Yes | Global | No |
| [**innodb\_ddl\_log\_crash\_reset\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ddl_log_crash_reset_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_disable\_sort\_file\_cache**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_disable_sort_file_cache) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_doublewrite\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_batch_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_doublewrite\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_files) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_fast\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fast_shutdown) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_flushing\_avg\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flushing_avg_loops) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_force\_load\_corrupted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_load_corrupted) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_fsync\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fsync_threshold) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_ft\_aux\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_aux_table) |  |  | Yes |  | Global | Yes |
| [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_ft\_enable\_diag\_print**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_diag_print) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword) | Yes | Yes | Yes |  | Both | Yes |
| [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_ft\_num\_word\_optimize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_num_word_optimize) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_ft\_result\_cache\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_result_cache_limit) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_ft\_sort\_pll\_degree**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_sort_pll_degree) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) | Yes | Yes | Yes |  | Both | Yes |
| [**Innodb\_have\_atomic\_builtins**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_have_atomic_builtins) |  |  |  | Yes | Global | No |
| [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_limit\_optimistic\_insert\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_limit_optimistic_insert_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) | Yes | Yes | Yes |  | Both | Yes |
| [**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) | Yes | Yes | Yes |  | Global | Varies |
| [**innodb\_log\_checkpoint\_fuzzy\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checkpoint_fuzzy_now) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_checkpoint\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checkpoint_now) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_checksums**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checksums) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_log\_spin\_cpu\_abs\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_spin_cpu_abs_lwm) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_spin\_cpu\_pct\_hwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_spin_cpu_pct_hwm) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_log\_wait\_for\_flush\_spin\_hwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_wait_for_flush_spin_hwm) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_log\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_log_waits) |  |  |  | Yes | Global | No |
| [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_log\_write\_requests**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_log_write_requests) |  |  |  | Yes | Global | No |
| [**innodb\_log\_writer\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_writer_threads) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_log\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_log_writes) |  |  |  | Yes | Global | No |
| [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_merge\_threshold\_set\_all\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_merge_threshold_set_all_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_monitor\_disable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_monitor\_reset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_monitor\_reset\_all**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset_all) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_num\_open\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_num_open_files) |  |  |  | Yes | Global | No |
| [**innodb\_numa\_interleave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_numa_interleave) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_open\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_open_files) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_os\_log\_fsyncs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_os_log_fsyncs) |  |  |  | Yes | Global | No |
| [**Innodb\_os\_log\_pending\_fsyncs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_os_log_pending_fsyncs) |  |  |  | Yes | Global | No |
| [**Innodb\_os\_log\_pending\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_os_log_pending_writes) |  |  |  | Yes | Global | No |
| [**Innodb\_os\_log\_written**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_os_log_written) |  |  |  | Yes | Global | No |
| [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) | Yes | Yes | Yes |  | Global | No |
| [**Innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_page_size) |  |  |  | Yes | Global | No |
| [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) | Yes | Yes | Yes |  | Global | No |
| [**Innodb\_pages\_created**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_pages_created) |  |  |  | Yes | Global | No |
| [**Innodb\_pages\_read**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_pages_read) |  |  |  | Yes | Global | No |
| [**Innodb\_pages\_written**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_pages_written) |  |  |  | Yes | Global | No |
| [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) | Yes | Yes | Yes |  | Session | Yes |
| [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_print\_ddl\_logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_ddl_logs) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_random\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_random_read_ahead) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_redo\_log\_archive\_dirs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_archive_dirs) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_redo\_log\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_redo_log_enabled) |  |  |  | Yes | Global | No |
| [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_replication\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_replication_delay) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_rollback\_on\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_row\_lock\_current\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_row_lock_current_waits) |  |  |  | Yes | Global | No |
| [**Innodb\_row\_lock\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_row_lock_time) |  |  |  | Yes | Global | No |
| [**Innodb\_row\_lock\_time\_avg**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_row_lock_time_avg) |  |  |  | Yes | Global | No |
| [**Innodb\_row\_lock\_time\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_row_lock_time_max) |  |  |  | Yes | Global | No |
| [**Innodb\_row\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_row_lock_waits) |  |  |  | Yes | Global | No |
| [**Innodb\_rows\_deleted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_rows_deleted) |  |  |  | Yes | Global | No |
| [**Innodb\_rows\_inserted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_rows_inserted) |  |  |  | Yes | Global | No |
| [**Innodb\_rows\_read**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_rows_read) |  |  |  | Yes | Global | No |
| [**Innodb\_rows\_updated**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_rows_updated) |  |  |  | Yes | Global | No |
| [**innodb\_saved\_page\_number\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_saved_page_number_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_segment\_reserve\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_segment_reserve_factor) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sort_buffer_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_method) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb-status-file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb-status-file) | Yes | Yes |  |  |  |  |
| [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) | Yes | Yes | Yes |  | Both | Yes |
| [**innodb\_sync\_array\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_array_size) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_sync\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_debug) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_sync\_spin\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_spin_loops) | Yes | Yes | Yes |  | Global | Yes |
| [**Innodb\_system\_rows\_deleted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_system_rows_deleted) |  |  |  | Yes | Global | No |
| [**Innodb\_system\_rows\_inserted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_system_rows_inserted) |  |  |  | Yes | Global | No |
| [**Innodb\_system\_rows\_read**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_system_rows_read) |  |  |  | Yes | Global | No |
| [**innodb\_table\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_table_locks) | Yes | Yes | Yes |  | Both | Yes |
| [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) | Yes | Yes | Yes |  | Both | Yes |
| [**Innodb\_truncated\_status\_writes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_truncated_status_writes) |  |  |  | Yes | Global | No |
| [**innodb\_trx\_purge\_view\_update\_only\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_trx_purge_view_update_only_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_trx\_rseg\_n\_slots\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_trx_rseg_n_slots_debug) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) | Yes | Yes | Yes |  | Global | Varies |
| [**Innodb\_undo\_tablespaces\_active**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_undo_tablespaces_active) |  |  |  | Yes | Global | No |
| [**Innodb\_undo\_tablespaces\_explicit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_undo_tablespaces_explicit) |  |  |  | Yes | Global | No |
| [**Innodb\_undo\_tablespaces\_implicit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_undo_tablespaces_implicit) |  |  |  | Yes | Global | No |
| [**Innodb\_undo\_tablespaces\_total**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_undo_tablespaces_total) |  |  |  | Yes | Global | No |
| [**innodb\_use\_fdatasync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_fdatasync) | Yes | Yes | Yes |  | Global | Yes |
| [**innodb\_use\_native\_aio**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_native_aio) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_validate\_tablespace\_paths**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_validate_tablespace_paths) | Yes | Yes | Yes |  | Global | No |
| [**innodb\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_version) |  |  | Yes |  | Global | No |
| [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) | Yes | Yes | Yes |  | Global | No |
| [**unique\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_unique_checks) |  |  | Yes |  | Both | Yes |

### InnoDB Command Options

[--innodb[=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "option_mysqld_innodb)***[value](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "option_mysqld_innodb)***[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "option_mysqld_innodb)

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb[=value]** |
| **Deprecated** | Yes |
| **Type** | Enumeration |
| **Default Value** | **ON** |
| **Valid Values** | **OFF**  **ON**  **FORCE** |

Controls loading of the **InnoDB** storage engine, if the server was compiled with **InnoDB** support. This option has a tristate format, with possible values of **OFF**, **ON**, or **FORCE**. See [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

To disable **InnoDB**, use [--innodb=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb) or [--skip-innodb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb). In this case, because the default storage engine is [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), the server does not start unless you also use [--default-storage-engine](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) and [--default-tmp-storage-engine](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_tmp_storage_engine) to set the default to some other engine for both permanent and **TEMPORARY** tables.

The **InnoDB** storage engine can no longer be disabled, and the [--innodb=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb) and [--skip-innodb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb) options are deprecated and have no effect. Their use results in a warning. Expect these options to be removed in a future MySQL release.

[--innodb-status-file](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "option_mysqld_innodb-status-file)

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-status-file[={OFF|ON}]** |
| **Type** | Boolean |
| **Default Value** | **OFF** |

The --innodb-status-file startup option controls whether **InnoDB** creates a file named innodb\_status.***pid*** in the data directory and writes [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output to it every 15 seconds, approximately.

The innodb\_status.***pid*** file is not created by default. To create it, start [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with the --innodb-status-file option. **InnoDB** removes the file when the server is shut down normally. If an abnormal shutdown occurs, the status file may have to be removed manually.

The --innodb-status-file option is intended for temporary use, as [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output generation can affect performance, and the innodb\_status.***pid*** file can become quite large over time.

For related information, see [Section 15.17.2, “Enabling InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-enabling-monitors).

[--skip-innodb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb)

Disable the **InnoDB** storage engine. See the description of [--innodb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb).

### InnoDB System Variables

**[daemon\_memcached\_enable\_binlog](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_enable_binlog)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-enable-binlog[={OFF|ON}]** |
| **System Variable** | [**daemon\_memcached\_enable\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_enable_binlog) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enable this option on the source server to use the **InnoDB** **memcached** plugin (**daemon\_memcached**) with the MySQL [binary log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_binary_log). This option can only be set at server startup. You must also enable the MySQL binary log on the source server using the [**--log-bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) option.

For more information, see [Section 15.20.7, “The InnoDB memcached Plugin and Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-replication).

**[daemon\_memcached\_engine\_lib\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_engine_lib_name)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-engine-lib-name=file\_name** |
| **System Variable** | [**daemon\_memcached\_engine\_lib\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_name) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **innodb\_engine.so** |

Specifies the shared library that implements the **InnoDB** **memcached** plugin.

For more information, see [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

**[daemon\_memcached\_engine\_lib\_path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_engine_lib_path)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-engine-lib-path=dir\_name** |
| **System Variable** | [**daemon\_memcached\_engine\_lib\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_path) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |
| **Default Value** | **NULL** |

The path of the directory containing the shared library that implements the **InnoDB** **memcached** plugin. The default value is NULL, representing the MySQL plugin directory. You should not need to modify this parameter unless specifying a **memcached** plugin for a different storage engine that is located outside of the MySQL plugin directory.

For more information, see [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

**[daemon\_memcached\_option](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_option)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-option=options** |
| **System Variable** | [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** |  |

Used to pass space-separated memcached options to the underlying **memcached** memory object caching daemon on startup. For example, you might change the port that **memcached** listens on, reduce the maximum number of simultaneous connections, change the maximum memory size for a key-value pair, or enable debugging messages for the error log.

See [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup) for usage details. For information about **memcached** options, refer to the **memcached** man page.

**[daemon\_memcached\_r\_batch\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_r_batch_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-r-batch-size=#** |
| **System Variable** | [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |

Specifies how many **memcached** read operations (**get** operations) to perform before doing a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to start a new transaction. Counterpart of [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size).

This value is set to 1 by default, so that any changes made to the table through SQL statements are immediately visible to **memcached** operations. You might increase it to reduce the overhead from frequent commits on a system where the underlying table is only being accessed through the **memcached** interface. If you set the value too large, the amount of undo or redo data could impose some storage overhead, as with any long-running transaction.

For more information, see [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

**[daemon\_memcached\_w\_batch\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_daemon_memcached_w_batch_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--daemon-memcached-w-batch-size=#** |
| **System Variable** | [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |

Specifies how many **memcached** write operations, such as **add**, **set**, and **incr**, to perform before doing a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to start a new transaction. Counterpart of [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size).

This value is set to 1 by default, on the assumption that data being stored is important to preserve in case of an outage and should immediately be committed. When storing non-critical data, you might increase this value to reduce the overhead from frequent commits; but then the last ***N***-1 uncommitted write operations could be lost if an unexpected exit occurs.

For more information, see [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

**[innodb\_adaptive\_flushing](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_adaptive_flushing)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-adaptive-flushing[={OFF|ON}]** |
| **System Variable** | [**innodb\_adaptive\_flushing**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies whether to dynamically adjust the rate of flushing [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) in the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) based on the workload. Adjusting the flush rate dynamically is intended to avoid bursts of I/O activity. This setting is enabled by default. See [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing) for more information. For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_adaptive\_flushing\_lwm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_adaptive_flushing_lwm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-adaptive-flushing-lwm=#** |
| **System Variable** | [**innodb\_adaptive\_flushing\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_flushing_lwm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10** |
| **Minimum Value** | **0** |
| **Maximum Value** | **70** |

Defines the low water mark representing percentage of [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) capacity at which [adaptive flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_adaptive_flushing) is enabled. For more information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

**[innodb\_adaptive\_hash\_index](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_adaptive_hash_index)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-adaptive-hash-index[={OFF|ON}]** |
| **System Variable** | [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Whether the **InnoDB** [adaptive hash index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_adaptive_hash_index) is enabled or disabled. It may be desirable, depending on your workload, to dynamically enable or disable [adaptive hash indexing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_adaptive_hash_index) to improve query performance. Because the adaptive hash index may not be useful for all workloads, conduct benchmarks with it both enabled and disabled, using realistic workloads. See [Section 15.5.3, “Adaptive Hash Index”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-adaptive-hash) for details.

This variable is enabled by default. You can modify this parameter using the **SET GLOBAL** statement, without restarting the server. Changing the setting at runtime requires privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges). You can also use --skip-innodb-adaptive-hash-index at server startup to disable it.

Disabling the adaptive hash index empties the hash table immediately. Normal operations can continue while the hash table is emptied, and executing queries that were using the hash table access the index B-trees directly instead. When the adaptive hash index is re-enabled, the hash table is populated again during normal operation.

**[innodb\_adaptive\_hash\_index\_parts](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_adaptive_hash_index_parts)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-adaptive-hash-index-parts=#** |
| **System Variable** | [**innodb\_adaptive\_hash\_index\_parts**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index_parts) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Numeric |
| **Default Value** | **8** |
| **Minimum Value** | **1** |
| **Maximum Value** | **512** |

Partitions the adaptive hash index search system. Each index is bound to a specific partition, with each partition protected by a separate latch.

The adaptive hash index search system is partitioned into 8 parts by default. The maximum setting is 512.

For related information, see [Section 15.5.3, “Adaptive Hash Index”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-adaptive-hash).

**[innodb\_adaptive\_max\_sleep\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_adaptive_max_sleep_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-adaptive-max-sleep-delay=#** |
| **System Variable** | [**innodb\_adaptive\_max\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_max_sleep_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **150000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000000** |

Permits **InnoDB** to automatically adjust the value of [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) up or down according to the current workload. Any nonzero value enables automated, dynamic adjustment of the [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) value, up to the maximum value specified in the [**innodb\_adaptive\_max\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_max_sleep_delay) option. The value represents the number of microseconds. This option can be useful in busy systems, with greater than 16 **InnoDB** threads. (In practice, it is most valuable for MySQL systems with hundreds or thousands of simultaneous connections.)

For more information, see [Section 15.8.4, “Configuring Thread Concurrency for InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-thread_concurrency).

**[innodb\_api\_bk\_commit\_interval](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_api_bk_commit_interval)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-api-bk-commit-interval=#** |
| **System Variable** | [**innodb\_api\_bk\_commit\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_bk_commit_interval) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **5** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1073741824** |

How often to auto-commit idle connections that use the **InnoDB** **memcached** interface, in seconds. For more information, see [Section 15.20.6.4, “Controlling Transactional Behavior of the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-txn).

**[innodb\_api\_disable\_rowlock](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_api_disable_rowlock)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-api-disable-rowlock[={OFF|ON}]** |
| **System Variable** | [**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Use this option to disable row locks when **InnoDB** **memcached** performs DML operations. By default, [**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) is disabled, which means that **memcached** requests row locks for **get** and **set** operations. When [**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) is enabled, **memcached** requests a table lock instead of row locks.

[**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) is not dynamic. It must be specified on the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) command line or entered in the MySQL configuration file. Configuration takes effect when the plugin is installed, which occurs when the MySQL server is started.

For more information, see [Section 15.20.6.4, “Controlling Transactional Behavior of the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-txn).

**[innodb\_api\_enable\_binlog](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_api_enable_binlog)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-api-enable-binlog[={OFF|ON}]** |
| **System Variable** | [**innodb\_api\_enable\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_binlog) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Lets you use the **InnoDB** **memcached** plugin with the MySQL [binary log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_binary_log). For more information, see [Enabling the InnoDB memcached Binary Log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-replication-enable-binlog).

**[innodb\_api\_enable\_mdl](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_api_enable_mdl)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-api-enable-mdl[={OFF|ON}]** |
| **System Variable** | [**innodb\_api\_enable\_mdl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_mdl) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Locks the table used by the **InnoDB** **memcached** plugin, so that it cannot be dropped or altered by [DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ddl) through the SQL interface. For more information, see [Section 15.20.6.4, “Controlling Transactional Behavior of the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-txn).

**[innodb\_api\_trx\_level](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_api_trx_level)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-api-trx-level=#** |
| **System Variable** | [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |

Controls the transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) on queries processed by the **memcached** interface. The constants corresponding to the familiar names are:

0 = [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted)

1 = [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed)

2 = [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read)

3 = [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable)

For more information, see [Section 15.20.6.4, “Controlling Transactional Behavior of the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-txn).

**[innodb\_autoextend\_increment](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_autoextend_increment)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-autoextend-increment=#** |
| **System Variable** | [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **64** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1000** |

The increment size (in megabytes) for extending the size of an auto-extending **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) file when it becomes full. The default value is 64. For related information, see [System Tablespace Data File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-data-file-configuration), and [Resizing the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-resize-system-tablespace).

The [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) setting does not affect [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace files or [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) files. These files are auto-extending regardless of the [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) setting. The initial extensions are by small amounts, after which extensions occur in increments of 4MB.

**[innodb\_autoinc\_lock\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_autoinc_lock_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-autoinc-lock-mode=#** |
| **System Variable** | [**innodb\_autoinc\_lock\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoinc_lock_mode) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2** |
| **Valid Values** | **0**  **1**  **2** |

The [lock mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lock_mode) to use for generating [auto-increment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_auto_increment) values. Permissible values are 0, 1, or 2, for traditional, consecutive, or interleaved, respectively.

The default setting is 2 (interleaved) as of MySQL 8.0, and 1 (consecutive) before that. The change to interleaved lock mode as the default setting reflects the change from statement-based to row-based replication as the default replication type, which occurred in MySQL 5.7. Statement-based replication requires the consecutive auto-increment lock mode to ensure that auto-increment values are assigned in a predictable and repeatable order for a given sequence of SQL statements, whereas row-based replication is not sensitive to the execution order of SQL statements.

For the characteristics of each lock mode, see [InnoDB AUTO\_INCREMENT Lock Modes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-lock-modes).

**[innodb\_background\_drop\_list\_empty](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_background_drop_list_empty)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-background-drop-list-empty[={OFF|ON}]** |
| **System Variable** | [**innodb\_background\_drop\_list\_empty**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_background_drop_list_empty) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enabling the [**innodb\_background\_drop\_list\_empty**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_background_drop_list_empty) debug option helps avoid test case failures by delaying table creation until the background drop list is empty. For example, if test case A places table **t1** on the background drop list, test case B waits until the background drop list is empty before creating table **t1**.

**[innodb\_buffer\_pool\_chunk\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_chunk_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-chunk-size=#** |
| **System Variable** | [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **134217728** |
| **Minimum Value** | **1048576** |
| **Maximum Value** | **innodb\_buffer\_pool\_size / innodb\_buffer\_pool\_instances** |

[**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) defines the chunk size for **InnoDB** buffer pool resizing operations.

To avoid copying all buffer pool pages during resizing operations, the operation is performed in “chunks”. By default, [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) is 128MB (134217728 bytes). The number of pages contained in a chunk depends on the value of [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size). [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) can be increased or decreased in units of 1MB (1048576 bytes).

The following conditions apply when altering the [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) value:

If [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) is larger than the current buffer pool size when the buffer pool is initialized, [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) is truncated to [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) / [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

Buffer pool size must always be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). If you alter [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is automatically rounded to a value that is equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). The adjustment occurs when the buffer pool is initialized.

**Important**

Care should be taken when changing [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), as changing this value can automatically increase the size of the buffer pool. Before changing [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size), calculate its effect on [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) to ensure that the resulting buffer pool size is acceptable.

To avoid potential performance issues, the number of chunks ([**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) / [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size)) should not exceed 1000.

The [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) variable is dynamic, which permits resizing the buffer pool while the server is online. However, the buffer pool size must be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances), and changing either of those variable settings requires restarting the server.

See [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize) for more information.

**[innodb\_buffer\_pool\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-debug[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_debug) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enabling this option permits multiple buffer pool instances when the buffer pool is less than 1GB in size, ignoring the 1GB minimum buffer pool size constraint imposed on [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). The [**innodb\_buffer\_pool\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_debug) option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_buffer\_pool\_dump\_at\_shutdown](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_dump_at_shutdown)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-dump-at-shutdown[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies whether to record the pages cached in the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) when the MySQL server is shut down, to shorten the [warmup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_warm_up) process at the next restart. Typically used in combination with [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup). The [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) option defines the percentage of most recently used buffer pool pages to dump.

Both [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) and [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) are enabled by default.

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_dump\_now](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_dump_now)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-dump-now[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_dump\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_now) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Immediately records the pages cached in the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). Typically used in combination with [**innodb\_buffer\_pool\_load\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_now).

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_dump\_pct](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_dump_pct)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-dump-pct=#** |
| **System Variable** | [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **25** |
| **Minimum Value** | **1** |
| **Maximum Value** | **100** |

Specifies the percentage of the most recently used pages for each buffer pool to read out and dump. The range is 1 to 100. The default value is 25. For example, if there are 4 buffer pools with 100 pages each, and [**innodb\_buffer\_pool\_dump\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_pct) is set to 25, the 25 most recently used pages from each buffer pool are dumped.

**[innodb\_buffer\_pool\_filename](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_filename)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-filename=file\_name** |
| **System Variable** | [**innodb\_buffer\_pool\_filename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_filename) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **ib\_buffer\_pool** |

Specifies the name of the file that holds the list of tablespace IDs and page IDs produced by [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) or [**innodb\_buffer\_pool\_dump\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_now). Tablespace IDs and page IDs are saved in the following format: **space, page\_id**. By default, the file is named ib\_buffer\_pool and is located in the **InnoDB** data directory. A non-default location must be specified relative to the data directory.

A file name can be specified at runtime, using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement:

SET GLOBAL innodb\_buffer\_pool\_filename=***'file\_name'***;

You can also specify a file name at startup, in a startup string or MySQL configuration file. When specifying a file name at startup, the file must exist or **InnoDB** returns a startup error indicating that there is no such file or directory.

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_in\_core\_file](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_in_core_file)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-in-core-file[={OFF|ON}]** |
| **Introduced** | 8.0.14 |
| **System Variable** | [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Disabling the [**innodb\_buffer\_pool\_in\_core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_in_core_file) variable reduces the size of core files by excluding **InnoDB** buffer pool pages. To use this variable, the [**core\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_core_file) variable must be enabled and the operating system must support the **MADV\_DONTDUMP** non-POSIX extension to **madvise()**, which is supported in Linux 3.4 and later. For more information, see [Section 15.8.3.7, “Excluding Buffer Pool Pages from Core Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-in-core-file).

**[innodb\_buffer\_pool\_instances](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_instances)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-instances=#** |
| **System Variable** | [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value (Other)** | **8 (or 1 if innodb\_buffer\_pool\_size < 1GB** |
| **Default Value (Windows, 32-bit platforms)** | **(autosized)** |
| **Minimum Value** | **1** |
| **Maximum Value** | **64** |

The number of regions that the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) is divided into. For systems with buffer pools in the multi-gigabyte range, dividing the buffer pool into separate instances can improve concurrency, by reducing contention as different threads read and write to cached pages. Each page that is stored in or read from the buffer pool is assigned to one of the buffer pool instances randomly, using a hashing function. Each buffer pool manages its own free lists, [flush lists](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush_list), [LRUs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lru), and all other data structures connected to a buffer pool, and is protected by its own buffer pool [mutex](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mutex).

This option only takes effect when setting [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) to 1GB or more. The total buffer pool size is divided among all the buffer pools. For best efficiency, specify a combination of [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) and [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) so that each buffer pool instance is at least 1GB.

The default value on 32-bit Windows systems depends on the value of [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size), as described below:

If [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is greater than 1.3GB, the default for [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) is [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size)/128MB, with individual memory allocation requests for each chunk. 1.3GB was chosen as the boundary at which there is significant risk for 32-bit Windows to be unable to allocate the contiguous address space needed for a single buffer pool.

Otherwise, the default is 1.

On all other platforms, the default value is 8 when [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) is greater than or equal to 1GB. Otherwise, the default is 1.

For related information, see [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize).

**[innodb\_buffer\_pool\_load\_abort](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_load_abort)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-load-abort[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_load\_abort**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_abort) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Interrupts the process of restoring **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) contents triggered by [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) or [**innodb\_buffer\_pool\_load\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_now).

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_load\_at\_startup](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_load_at_startup)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-load-at-startup[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies that, on MySQL server startup, the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) is automatically [warmed up](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_warm_up) by loading the same pages it held at an earlier time. Typically used in combination with [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown).

Both [**innodb\_buffer\_pool\_dump\_at\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_dump_at_shutdown) and [**innodb\_buffer\_pool\_load\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_at_startup) are enabled by default.

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_load\_now](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_load_now)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-load-now[={OFF|ON}]** |
| **System Variable** | [**innodb\_buffer\_pool\_load\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_load_now) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Immediately [warms up](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_warm_up) the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) by loading a set of data pages, without waiting for a server restart. Can be useful to bring cache memory back to a known state during benchmarking, or to ready the MySQL server to resume its normal workload after running queries for reports or maintenance.

For more information, see [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

**[innodb\_buffer\_pool\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_buffer_pool_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-buffer-pool-size=#** |
| **System Variable** | [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **134217728** |
| **Minimum Value** | **5242880** |
| **Maximum Value (64-bit platforms)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms)** | **2\*\*32-1** |

The size in bytes of the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), the memory area where **InnoDB** caches table and index data. The default value is 134217728 bytes (128MB). The maximum value depends on the CPU architecture; the maximum is 4294967295 (232-1) on 32-bit systems and 18446744073709551615 (264-1) on 64-bit systems. On 32-bit systems, the CPU architecture and operating system may impose a lower practical maximum size than the stated maximum. When the size of the buffer pool is greater than 1GB, setting [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) to a value greater than 1 can improve the scalability on a busy server.

A larger buffer pool requires less disk I/O to access the same table data more than once. On a dedicated database server, you might set the buffer pool size to 80% of the machine's physical memory size. Be aware of the following potential issues when configuring buffer pool size, and be prepared to scale back the size of the buffer pool if necessary.

Competition for physical memory can cause paging in the operating system.

**InnoDB** reserves additional memory for buffers and control structures, so that the total allocated space is approximately 10% greater than the specified buffer pool size.

Address space for the buffer pool must be contiguous, which can be an issue on Windows systems with DLLs that load at specific addresses.

The time to initialize the buffer pool is roughly proportional to its size. On instances with large buffer pools, initialization time might be significant. To reduce the initialization period, you can save the buffer pool state at server shutdown and restore it at server startup. See [Section 15.8.3.6, “Saving and Restoring the Buffer Pool State”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-preload-buffer-pool).

When you increase or decrease buffer pool size, the operation is performed in chunks. Chunk size is defined by the [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) variable, which has a default of 128 MB.

Buffer pool size must always be equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances). If you alter the buffer pool size to a value that is not equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances), buffer pool size is automatically adjusted to a value that is equal to or a multiple of [**innodb\_buffer\_pool\_chunk\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_chunk_size) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

[**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) can be set dynamically, which allows you to resize the buffer pool without restarting the server. The [**Innodb\_buffer\_pool\_resize\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_resize_status) status variable reports the status of online buffer pool resizing operations. See [Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-resize) for more information.

If [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled, the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) value is automatically configured if it is not explicitly defined. For more information, see [Section 15.8.12, “Enabling Automatic Configuration for a Dedicated MySQL Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-dedicated-server).

**[innodb\_change\_buffer\_max\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_change_buffer_max_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-change-buffer-max-size=#** |
| **System Variable** | [**innodb\_change\_buffer\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffer_max_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **25** |
| **Minimum Value** | **0** |
| **Maximum Value** | **50** |

Maximum size for the **InnoDB** [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer), as a percentage of the total size of the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). You might increase this value for a MySQL server with heavy insert, update, and delete activity, or decrease it for a MySQL server with unchanging data used for reporting. For more information, see [Section 15.5.2, “Change Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_change\_buffering](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_change_buffering)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-change-buffering=value** |
| **System Variable** | [**innodb\_change\_buffering**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **all** |
| **Valid Values** | **none**  **inserts**  **deletes**  **changes**  **purges**  **all** |

Whether **InnoDB** performs [change buffering](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffering), an optimization that delays write operations to secondary indexes so that the I/O operations can be performed sequentially. Permitted values are described in the following table. Values may also be specified numerically.

**Table 15.25 Permitted Values for innodb\_change\_buffering**

| **Value** | **Numeric Value** | **Description** |
| --- | --- | --- |
| **none** | **0** | Do not buffer any operations. |
| **inserts** | **1** | Buffer insert operations. |
| **deletes** | **2** | Buffer delete marking operations; strictly speaking, the writes that mark index records for later deletion during a purge operation. |
| **changes** | **3** | Buffer inserts and delete-marking operations. |
| **purges** | **4** | Buffer the physical deletion operations that happen in the background. |
| **all** | **5** | The default. Buffer inserts, delete-marking operations, and purges. |

For more information, see [Section 15.5.2, “Change Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_change\_buffering\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_change_buffering_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-change-buffering-debug=#** |
| **System Variable** | [**innodb\_change\_buffering\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_change_buffering_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Maximum Value** | **2** |

Sets a debug flag for **InnoDB** change buffering. A value of 1 forces all changes to the change buffer. A value of 2 causes an unexpected exit at merge. A default value of 0 indicates that the change buffering debug flag is not set. This option is only available when debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_checkpoint\_disabled](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_checkpoint_disabled)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-checkpoint-disabled[={OFF|ON}]** |
| **System Variable** | [**innodb\_checkpoint\_disabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checkpoint_disabled) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This is a debug option that is only intended for expert debugging use. It disables checkpoints so that a deliberate server exit always initiates **InnoDB** recovery. It should only be enabled for a short interval, typically before running DML operations that write redo log entries that would require recovery following a server exit. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_checksum\_algorithm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_checksum_algorithm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-checksum-algorithm=value** |
| **System Variable** | [**innodb\_checksum\_algorithm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checksum_algorithm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **crc32** |
| **Valid Values** | **innodb**  **crc32**  **none**  **strict\_innodb**  **strict\_crc32**  **strict\_none** |

Specifies how to generate and verify the [checksum](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checksum) stored in the disk blocks of **InnoDB** [tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_tablespace). The default value for [**innodb\_checksum\_algorithm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checksum_algorithm) is **crc32**.

Versions of [MySQL Enterprise Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) up to 3.8.0 do not support backing up tablespaces that use CRC32 checksums. [MySQL Enterprise Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) adds CRC32 checksum support in 3.8.1, with some limitations. Refer to the [MySQL Enterprise Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) 3.8.1 Change History for more information.

The value **innodb** is backward-compatible with earlier versions of MySQL. The value **crc32** uses an algorithm that is faster to compute the checksum for every modified block, and to check the checksums for each disk read. It scans blocks 32 bits at a time, which is faster than the **innodb** checksum algorithm, which scans blocks 8 bits at a time. The value **none** writes a constant value in the checksum field rather than computing a value based on the block data. The blocks in a tablespace can use a mix of old, new, and no checksum values, being updated gradually as the data is modified; once blocks in a tablespace are modified to use the **crc32** algorithm, the associated tables cannot be read by earlier versions of MySQL.

The strict form of a checksum algorithm reports an error if it encounters a valid but non-matching checksum value in a tablespace. It is recommended that you only use strict settings in a new instance, to set up tablespaces for the first time. Strict settings are somewhat faster, because they do not need to compute all checksum values during disk reads.

The following table shows the difference between the **none**, **innodb**, and **crc32** option values, and their strict counterparts. **none**, **innodb**, and **crc32** write the specified type of checksum value into each data block, but for compatibility accept other checksum values when verifying a block during a read operation. Strict settings also accept valid checksum values but print an error message when a valid non-matching checksum value is encountered. Using the strict form can make verification faster if all **InnoDB** data files in an instance are created under an identical [**innodb\_checksum\_algorithm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_checksum_algorithm) value.

**Table 15.26 Permitted innodb\_checksum\_algorithm Values**

| **Value** | **Generated checksum (when writing)** | **Permitted checksums (when reading)** |
| --- | --- | --- |
| **none** | A constant number. | Any of the checksums generated by **none**, **innodb**, or **crc32**. |
| **innodb** | A checksum calculated in software, using the original algorithm from **InnoDB**. | Any of the checksums generated by **none**, **innodb**, or **crc32**. |
| **crc32** | A checksum calculated using the **crc32** algorithm, possibly done with a hardware assist. | Any of the checksums generated by **none**, **innodb**, or **crc32**. |
| **strict\_none** | A constant number | Any of the checksums generated by **none**, **innodb**, or **crc32**. **InnoDB** prints an error message if a valid but non-matching checksum is encountered. |
| **strict\_innodb** | A checksum calculated in software, using the original algorithm from **InnoDB**. | Any of the checksums generated by **none**, **innodb**, or **crc32**. **InnoDB** prints an error message if a valid but non-matching checksum is encountered. |
| **strict\_crc32** | A checksum calculated using the **crc32** algorithm, possibly done with a hardware assist. | Any of the checksums generated by **none**, **innodb**, or **crc32**. **InnoDB** prints an error message if a valid but non-matching checksum is encountered. |

**[innodb\_cmp\_per\_index\_enabled](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_cmp_per_index_enabled)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-cmp-per-index-enabled[={OFF|ON}]** |
| **System Variable** | [**innodb\_cmp\_per\_index\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_cmp_per_index_enabled) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables per-index compression-related statistics in the [**INFORMATION\_SCHEMA.INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table) table. Because these statistics can be expensive to gather, only enable this option on development, test, or replica instances during performance tuning related to **InnoDB** [compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) tables.

For more information, see [Section 26.4.8, “The INFORMATION\_SCHEMA INNODB\_CMP\_PER\_INDEX and INNODB\_CMP\_PER\_INDEX\_RESET Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table), and [Section 15.9.1.4, “Monitoring InnoDB Table Compression at Runtime”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-tuning-monitoring).

**[innodb\_commit\_concurrency](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_commit_concurrency)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-commit-concurrency=#** |
| **System Variable** | [**innodb\_commit\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_commit_concurrency) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000** |

The number of [threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_thread) that can [commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit) at the same time. A value of 0 (the default) permits any number of [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction) to commit simultaneously.

The value of [**innodb\_commit\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_commit_concurrency) cannot be changed at runtime from zero to nonzero or vice versa. The value can be changed from one nonzero value to another.

**[innodb\_compress\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_compress_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-compress-debug=value** |
| **System Variable** | [**innodb\_compress\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compress_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **none** |
| **Valid Values** | **none**  **zlib**  **lz4**  **lz4hc** |

Compresses all tables using a specified compression algorithm without having to define a **COMPRESSION** attribute for each table. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

For related information, see [Section 15.9.2, “InnoDB Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-page-compression).

**[innodb\_compression\_failure\_threshold\_pct](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_compression_failure_threshold_pct)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-compression-failure-threshold-pct=#** |
| **System Variable** | [**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **5** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |

Defines the compression failure rate threshold for a table, as a percentage, at which point MySQL begins adding padding within [compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) pages to avoid expensive [compression failures](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure). When this threshold is passed, MySQL begins to leave additional free space within each new compressed page, dynamically adjusting the amount of free space up to the percentage of page size specified by [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max). A value of zero disables the mechanism that monitors compression efficiency and dynamically adjusts the padding amount.

For more information, see [Section 15.9.1.6, “Compression for OLTP Workloads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp).

**[innodb\_compression\_level](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_compression_level)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-compression-level=#** |
| **System Variable** | [**innodb\_compression\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_level) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **6** |
| **Minimum Value** | **0** |
| **Maximum Value** | **9** |

Specifies the level of zlib compression to use for **InnoDB** [compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) tables and indexes. A higher value lets you fit more data onto a storage device, at the expense of more CPU overhead during compression. A lower value lets you reduce CPU overhead when storage space is not critical, or you expect the data is not especially compressible.

For more information, see [Section 15.9.1.6, “Compression for OLTP Workloads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp).

**[innodb\_compression\_pad\_pct\_max](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_compression_pad_pct_max)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-compression-pad-pct-max=#** |
| **System Variable** | [**innodb\_compression\_pad\_pct\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_pad_pct_max) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **0** |
| **Maximum Value** | **75** |

Specifies the maximum percentage that can be reserved as free space within each compressed [page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page), allowing room to reorganize the data and modification log within the page when a [compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) table or index is updated and the data might be recompressed. Only applies when [**innodb\_compression\_failure\_threshold\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_compression_failure_threshold_pct) is set to a nonzero value, and the rate of [compression failures](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression_failure) passes the cutoff point.

For more information, see [Section 15.9.1.6, “Compression for OLTP Workloads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp).

**[innodb\_concurrency\_tickets](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_concurrency_tickets)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-concurrency-tickets=#** |
| **System Variable** | [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **5000** |
| **Minimum Value** | **1** |
| **Maximum Value** | **4294967295** |

Determines the number of [threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_thread) that can enter **InnoDB** concurrently. A thread is placed in a queue when it tries to enter **InnoDB** if the number of threads has already reached the concurrency limit. When a thread is permitted to enter **InnoDB**, it is given a number of “ tickets” equal to the value of [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets), and the thread can enter and leave **InnoDB** freely until it has used up its tickets. After that point, the thread again becomes subject to the concurrency check (and possible queuing) the next time it tries to enter **InnoDB**. The default value is 5000.

With a small [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) value, small transactions that only need to process a few rows compete fairly with larger transactions that process many rows. The disadvantage of a small [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) value is that large transactions must loop through the queue many times before they can complete, which extends the amount of time required to complete their task.

With a large [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) value, large transactions spend less time waiting for a position at the end of the queue (controlled by [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency)) and more time retrieving rows. Large transactions also require fewer trips through the queue to complete their task. The disadvantage of a large [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) value is that too many large transactions running at the same time can starve smaller transactions by making them wait a longer time before executing.

With a nonzero [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) value, you may need to adjust the [**innodb\_concurrency\_tickets**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_concurrency_tickets) value up or down to find the optimal balance between larger and smaller transactions. The **SHOW ENGINE INNODB STATUS** report shows the number of tickets remaining for an executing transaction in its current pass through the queue. This data may also be obtained from the **TRX\_CONCURRENCY\_TICKETS** column of the [**INFORMATION\_SCHEMA.INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table.

For more information, see [Section 15.8.4, “Configuring Thread Concurrency for InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-thread_concurrency).

**[innodb\_data\_file\_path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_data_file_path)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-data-file-path=file\_name** |
| **System Variable** | [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **ibdata1:12M:autoextend** |

Defines the name, size, and attributes of **InnoDB** system tablespace data files. If you do not specify a value for [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path), the default behavior is to create a single auto-extending data file, slightly larger than 12MB, named ibdata1.

The full syntax for a data file specification includes the file name, file size, **autoextend** attribute, and **max** attribute:

***file\_name***:***file\_size***[:autoextend[:max:***max\_file\_size***]]

File sizes are specified in kilobytes, megabytes, or gigabytes by appending **K**, **M** or **G** to the size value. If specifying the data file size in kilobytes, do so in multiples of 1024. Otherwise, KB values are rounded to nearest megabyte (MB) boundary. The sum of file sizes must be, at a minimum, slightly larger than 12MB.

For additional configuration information, see [System Tablespace Data File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-data-file-configuration). For resizing instructions, see [Resizing the System Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-resize-system-tablespace).

**[innodb\_data\_home\_dir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_data_home_dir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-data-home-dir=dir\_name** |
| **System Variable** | [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |

The common part of the directory path for **InnoDB** [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) data files. The default value is the MySQL data directory. The setting is concatenated with the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) setting, unless that setting is defined with an absolute path.

A trailing slash is required when specifying a value for [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir). For example:

[mysqld]

innodb\_data\_home\_dir = /path/to/myibdata/

This setting does not affect the location of [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces.

For related information, see [Section 15.8.1, “InnoDB Startup Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-init-startup-configuration).

**[innodb\_ddl\_log\_crash\_reset\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ddl_log_crash_reset_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ddl-log-crash-reset-debug[={OFF|ON}]** |
| **System Variable** | [**innodb\_ddl\_log\_crash\_reset\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ddl_log_crash_reset_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enable this debug option to reset DDL log crash injection counters to 1. This option is only available when debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_deadlock\_detect](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_deadlock_detect)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-deadlock-detect[={OFF|ON}]** |
| **System Variable** | [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

This option is used to disable deadlock detection. On high concurrency systems, deadlock detection can cause a slowdown when numerous threads wait for the same lock. At times, it may be more efficient to disable deadlock detection and rely on the [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) setting for transaction rollback when a deadlock occurs.

For related information, see [Section 15.7.5.2, “Deadlock Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection).

**[innodb\_dedicated\_server](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_dedicated_server)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-dedicated-server[={OFF|ON}]** |
| **System Variable** | [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

When [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled, **InnoDB** automatically configures the following variables:

[**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size)

[**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size)

[**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) (as of MySQL 8.0.14)

[**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method)

Only consider enabling [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) if the MySQL instance resides on a dedicated server where it can use all available system resources. Enabling [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is not recommended if the MySQL instance shares system resources with other applications.

For more information, see [Section 15.8.12, “Enabling Automatic Configuration for a Dedicated MySQL Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-dedicated-server).

**[innodb\_default\_row\_format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_default_row_format)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-default-row-format=value** |
| **System Variable** | [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **DYNAMIC** |
| **Valid Values** | **DYNAMIC**  **COMPACT**  **REDUNDANT** |

The [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) option defines the default row format for **InnoDB** tables and user-created temporary tables. The default setting is **DYNAMIC**. Other permitted values are **COMPACT** and **REDUNDANT**. The **COMPRESSED** row format, which is not supported for use in the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace), cannot be defined as the default.

Newly created tables use the row format defined by [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) when a **ROW\_FORMAT** option is not specified explicitly or when **ROW\_FORMAT=DEFAULT** is used.

When a **ROW\_FORMAT** option is not specified explicitly or when **ROW\_FORMAT=DEFAULT** is used, any operation that rebuilds a table also silently changes the row format of the table to the format defined by [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format). For more information, see [Defining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-defining).

Internal **InnoDB** temporary tables created by the server to process queries use the **DYNAMIC** row format, regardless of the [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format) setting.

**[innodb\_directories](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_directories)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-directories=dir\_name** |
| **System Variable** | [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |
| **Default Value** | **NULL** |

Defines directories to scan at startup for tablespace files. This option is used when moving or restoring tablespace files to a new location while the server is offline. It is also used to specify directories of tablespace files created using an absolute path or that reside outside of the data directory.

Tablespace discovery during crash recovery relies on the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting to identify tablespaces referenced in the redo logs. For more information, see [Tablespace Discovery During Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-tablespace-discovery).

The default value is NULL, but directories defined by [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) are always appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value when **InnoDB** builds a list of directories to scan at startup. These directories are appended regardless of whether an [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting is specified explicitly.

[**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) may be specified as an option in a startup command or in a MySQL option file. Quotes surround the argument value because otherwise some command interpreters interpret semicolon (**;**) as a special character. (For example, Unix shells treat it as a command terminator.)

Startup command:

mysqld --innodb-directories="***directory\_path\_1***;***directory\_path\_2***"

MySQL option file:

[mysqld]

innodb\_directories="***directory\_path\_1***;***directory\_path\_2***"

Wildcard expressions cannot be used to specify directories.

The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) scan also traverses the subdirectories of specified directories. Duplicate directories and subdirectories are discarded from the list of directories to be scanned.

For more information, see [Section 15.6.3.6, “Moving Tablespace Files While the Server is Offline”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-moving-data-files-offline).

**[innodb\_disable\_sort\_file\_cache](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_disable_sort_file_cache)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-disable-sort-file-cache[={OFF|ON}]** |
| **System Variable** | [**innodb\_disable\_sort\_file\_cache**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_disable_sort_file_cache) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Disables the operating system file system cache for merge-sort temporary files. The effect is to open such files with the equivalent of **O\_DIRECT**.

**[innodb\_doublewrite](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_doublewrite)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-doublewrite[={OFF|ON}]** |
| **System Variable** | [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

The [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) variable controls whether the doublwrite buffer is enabled. It is enabled by default in most cases. To disable the doublewrite buffer, set [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) to 0 or start the server with **--skip-innodb-doublewrite**. You might consider disabling the doublewrite buffer if you are more concerned with performance than data integrity, as may be the case when performing benchmarks, for example.

If the doublewrite buffer is located on a Fusion-io device that supports atomic writes, the doublewrite buffer is automatically disabled and data file writes are performed using Fusion-io atomic writes instead. However, be aware that the [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) setting is global. When the doublewrite buffer is disabled, it is disabled for all data files including those that do not reside on Fusion-io hardware. This feature is only supported on Fusion-io hardware and is only enabled for Fusion-io NVMFS on Linux. To take full advantage of this feature, an [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) setting of **O\_DIRECT** is recommended.

For related information, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**[innodb\_doublewrite\_batch\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_doublewrite_batch_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-doublewrite-batch-size=#** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**innodb\_doublewrite\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_batch_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **256** |

Defines the number of doublewrite pages to write in a batch.

For more information, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**[innodb\_doublewrite\_dir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_doublewrite_dir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-doublewrite-dir=dir\_name** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**innodb\_doublewrite\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_dir) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |

Defines the directory for doublewrite files. If no directory is specified, doublewrite files are created in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory, which defaults to the data directory if unspecified.

For more information, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**[innodb\_doublewrite\_files](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_doublewrite_files)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-doublewrite-files=#** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**innodb\_doublewrite\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_files) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **innodb\_buffer\_pool\_instances \* 2** |
| **Minimum Value** | **2** |
| **Maximum Value** | **256** |

Defines the number of doublewrite files. By default, two doublewrite files are created for each buffer pool instance.

At a minimum, there are two doublewrite files. The maximum number of doublewrite files is two times the number of buffer pool instances. (The number of buffer pool instances is controlled by the [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) variable.)

For more information, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**[innodb\_doublewrite\_pages](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_doublewrite_pages)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-doublewrite-pages=#** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **innodb\_write\_io\_threads value** |
| **Minimum Value** | **innodb\_write\_io\_threads value** |
| **Maximum Value** | **512** |

Defines the maximum number of doublewrite pages per thread for a batch write. If no value is specified, [**innodb\_doublewrite\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite_pages) is set to the [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) value.

For more information, see [Section 15.6.4, “Doublewrite Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-doublewrite-buffer).

**[innodb\_extend\_and\_initialize](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_extend_and_initialize)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb=extend-and-initialize[={OFF|ON}]** |
| **Introduced** | 8.0.22 |
| **System Variable** | [**innodb\_extend\_and\_initialize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_extend_and_initialize) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Controls how space is allocated to file-per-table and general tablespaces on Linux systems.

When enabled, **InnoDB** writes NULLs to newly allocated pages. When disabled, space is allocated using **posix\_fallocate()** calls, which reserve space without physically writing NULLs.

For more information, see [Section 15.6.3.8, “Optimizing Tablespace Space Allocation on Linux”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-optimize-tablespace-page-allocation).

**[innodb\_fast\_shutdown](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_fast_shutdown)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-fast-shutdown=#** |
| **System Variable** | [**innodb\_fast\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fast_shutdown) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Valid Values** | **0**  **1**  **2** |

The **InnoDB** [shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_shutdown) mode. If the value is 0, **InnoDB** does a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown), a full [purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge) and a change buffer merge before shutting down. If the value is 1 (the default), **InnoDB** skips these operations at shutdown, a process known as a [fast shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fast_shutdown). If the value is 2, **InnoDB** flushes its logs and shuts down cold, as if MySQL had crashed; no committed transactions are lost, but the [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) operation makes the next startup take longer.

The slow shutdown can take minutes, or even hours in extreme cases where substantial amounts of data are still buffered. Use the slow shutdown technique before upgrading or downgrading between MySQL major releases, so that all data files are fully prepared in case the upgrade process updates the file format.

Use [**innodb\_fast\_shutdown=2**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fast_shutdown) in emergency or troubleshooting situations, to get the absolute fastest shutdown if data is at risk of corruption.

**[innodb\_fil\_make\_page\_dirty\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_fil_make_page_dirty_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-fil-make-page-dirty-debug=#** |
| **System Variable** | [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Maximum Value** | **2\*\*32-1** |

By default, setting [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) to the ID of a tablespace immediately dirties the first page of the tablespace. If [**innodb\_saved\_page\_number\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_saved_page_number_debug) is set to a non-default value, setting [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) dirties the specified page. The [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_file\_per\_table](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_file_per_table)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-file-per-table[={OFF|ON}]** |
| **System Variable** | [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

When [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is enabled, tables are created in file-per-table tablespaces by default. When disabled, tables are created in the system tablespace by default. For information about file-per-table tablespaces, see [Section 15.6.3.2, “File-Per-Table Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-per-table-tablespaces). For information about the **InnoDB** system tablespace, see [Section 15.6.3.1, “The System Tablespace”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-system-tablespace).

The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable can be configured at runtime using a [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement, specified on the command line at startup, or specified in an option file. Configuration at runtime requires privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)) and immediately affects the operation of all connections.

When a table that resides in a file-per-table tablespace is truncated or dropped, the freed space is returned to the operating system. Truncating or dropping a table that resides in the system tablespace only frees space in the system tablespace. Freed space in the system tablespace can be used again for **InnoDB** data but is not returned to the operating system, as system tablespace data files never shrink.

The [**innodb\_file\_per-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting does not affect the creation of temporary tables. As of MySQL 8.0.14, temporary tables are created in session temporary tablespaces, and in the global temporary tablespace before that. See [Section 15.6.3.5, “Temporary Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-temporary-tablespace).

**[innodb\_fill\_factor](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_fill_factor)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-fill-factor=#** |
| **System Variable** | [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **100** |
| **Minimum Value** | **10** |
| **Maximum Value** | **100** |

**InnoDB** performs a bulk load when creating or rebuilding indexes. This method of index creation is known as a “sorted index build”.

[**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) defines the percentage of space on each B-tree page that is filled during a sorted index build, with the remaining space reserved for future index growth. For example, setting [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) to 80 reserves 20 percent of the space on each B-tree page for future index growth. Actual percentages may vary. The [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) setting is interpreted as a hint rather than a hard limit.

An [**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) setting of 100 leaves 1/16 of the space in clustered index pages free for future index growth.

[**innodb\_fill\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fill_factor) applies to both B-tree leaf and non-leaf pages. It does not apply to external pages used for [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) entries.

For more information, see [Section 15.6.2.3, “Sorted Index Builds”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sorted-index-builds).

**[innodb\_flush\_log\_at\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flush_log_at_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flush-log-at-timeout=#** |
| **System Variable** | [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **2700** |

Write and flush the logs every ***N*** seconds. [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout) allows the timeout period between flushes to be increased in order to reduce flushing and avoid impacting performance of binary log group commit. The default setting for [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout) is once per second.

**[innodb\_flush\_log\_at\_trx\_commit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flush_log_at_trx_commit)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flush-log-at-trx-commit=#** |
| **System Variable** | [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **1** |
| **Valid Values** | **0**  **1**  **2** |

Controls the balance between strict [ACID](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_acid) compliance for [commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit) operations and higher performance that is possible when commit-related I/O operations are rearranged and done in batches. You can achieve better performance by changing the default value but then you can lose transactions in a crash.

The default setting of 1 is required for full ACID compliance. Logs are written and flushed to disk at each transaction commit.

With a setting of 0, logs are written and flushed to disk once per second. Transactions for which logs have not been flushed can be lost in a crash.

With a setting of 2, logs are written after each transaction commit and flushed to disk once per second. Transactions for which logs have not been flushed can be lost in a crash.

For settings 0 and 2, once-per-second flushing is not 100% guaranteed. Flushing may occur more frequently due to DDL changes and other internal **InnoDB** activities that cause logs to be flushed independently of the [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) setting, and sometimes less frequently due to scheduling issues. If logs are flushed once per second, up to one second of transactions can be lost in a crash. If logs are flushed more or less frequently than once per second, the amount of transactions that can be lost varies accordingly.

Log flushing frequency is controlled by [**innodb\_flush\_log\_at\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_timeout), which allows you to set log flushing frequency to ***N*** seconds (where ***N*** is **1 ... 2700**, with a default value of 1). However, any unexpected [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process exit can erase up to ***N*** seconds of transactions.

DDL changes and other internal **InnoDB** activities flush the log independently of the [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) setting.

**InnoDB** [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) works regardless of the [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) setting. Transactions are either applied entirely or erased entirely.

For durability and consistency in a replication setup that uses **InnoDB** with transactions:

If binary logging is enabled, set **sync\_binlog=1**.

Always set [**innodb\_flush\_log\_at\_trx\_commit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit).

For information on the combination of settings on a replica that is most resilient to unexpected halts, see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

**Caution**

Many operating systems and some disk hardware fool the flush-to-disk operation. They may tell [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) that the flush has taken place, even though it has not. In this case, the durability of transactions is not guaranteed even with the recommended settings, and in the worst case, a power outage can corrupt **InnoDB** data. Using a battery-backed disk cache in the SCSI disk controller or in the disk itself speeds up file flushes, and makes the operation safer. You can also try to disable the caching of disk writes in hardware caches.

**[innodb\_flush\_method](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flush_method)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flush-method=value** |
| **System Variable** | [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value (Windows)** | **unbuffered** |
| **Default Value (Unix)** | **fsync** |
| **Valid Values (Windows)** | **unbuffered**  **normal** |
| **Valid Values (Unix)** | **fsync**  **O\_DSYNC**  **littlesync**  **nosync**  **O\_DIRECT**  **O\_DIRECT\_NO\_FSYNC** |

Defines the method used to [flush](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) data to **InnoDB** [data files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_files) and [log files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file), which can affect I/O throughput.

On Unix-like systems, the default value is **fsync**. On Windows, the default value is **unbuffered**.

**Note**

In MySQL 8.0, [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) options may be specified numerically.

The [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) options for Unix-like systems include:

**fsync** or **0**: **InnoDB** uses the **fsync()** system call to flush both the data and log files. **fsync** is the default setting.

**O\_DSYNC** or **1**: **InnoDB** uses **O\_SYNC** to open and flush the log files, and **fsync()** to flush the data files. **InnoDB** does not use **O\_DSYNC** directly because there have been problems with it on many varieties of Unix.

**littlesync** or **2**: This option is used for internal performance testing and is currently unsupported. Use at your own risk.

**nosync** or **3**: This option is used for internal performance testing and is currently unsupported. Use at your own risk.

**O\_DIRECT** or **4**: **InnoDB** uses **O\_DIRECT** (or **directio()** on Solaris) to open the data files, and uses **fsync()** to flush both the data and log files. This option is available on some GNU/Linux versions, FreeBSD, and Solaris.

**O\_DIRECT\_NO\_FSYNC**: **InnoDB** uses **O\_DIRECT** during flushing I/O, but skips the **fsync()** system call after each write operation.

Prior to MySQL 8.0.14, this setting is not suitable for file systems such as XFS and EXT4, which require an **fsync()** system call to synchronize file system metadata changes. If you are not sure whether your file system requires an **fsync()** system call to synchronize file system metadata changes, use **O\_DIRECT** instead.

As of MySQL 8.0.14, **fsync()** is called after creating a new file, after increasing file size, and after closing a file, to ensure that file system metadata changes are synchronized. The **fsync()** system call is still skipped after each write operation.

Data loss is possible if redo log files and data files reside on different storage devices, and an unexpected exit occurs before data file writes are flushed from a device cache that is not battery-backed. If you use or intend to use different storage devices for redo log files and data files, and your data files reside on a device with a cache that is not battery-backed, use **O\_DIRECT** instead.

The [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) options for Windows systems include:

**unbuffered** or **0**: **InnoDB** uses simulated asynchronous I/O and non-buffered I/O.

**normal** or **1**: **InnoDB** uses simulated asynchronous I/O and buffered I/O.

How each setting affects performance depends on hardware configuration and workload. Benchmark your particular configuration to decide which setting to use, or whether to keep the default setting. Examine the [**Innodb\_data\_fsyncs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_data_fsyncs) status variable to see the overall number of **fsync()** calls for each setting. The mix of read and write operations in your workload can affect how a setting performs. For example, on a system with a hardware RAID controller and battery-backed write cache, **O\_DIRECT** can help to avoid double buffering between the **InnoDB** buffer pool and the operating system file system cache. On some systems where **InnoDB** data and log files are located on a SAN, the default value or **O\_DSYNC** might be faster for a read-heavy workload with mostly **SELECT** statements. Always test this parameter with hardware and workload that reflect your production environment. For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

If [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled, the [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) value is automatically configured if it is not explicitly defined. For more information, see [Section 15.8.12, “Enabling Automatic Configuration for a Dedicated MySQL Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-dedicated-server).

**[innodb\_flush\_neighbors](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flush_neighbors)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flush-neighbors=#** |
| **System Variable** | [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **0** |
| **Valid Values** | **0**  **1**  **2** |

Specifies whether [flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) a page from the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) also flushes other [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) in the same [extent](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_extent).

A setting of 0 disables [**innodb\_flush\_neighbors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_neighbors). Dirty pages in the same extent are not flushed.

A setting of 1 flushes contiguous dirty pages in the same extent.

A setting of 2 flushes dirty pages in the same extent.

When the table data is stored on a traditional [HDD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_hdd) storage device, flushing such [neighbor pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_neighbor_page) in one operation reduces I/O overhead (primarily for disk seek operations) compared to flushing individual pages at different times. For table data stored on [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd), seek time is not a significant factor and you can set this option to 0 to spread out write operations. For related information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

**[innodb\_flush\_sync](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flush_sync)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flush-sync[={OFF|ON}]** |
| **System Variable** | [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

The [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) variable, which is enabled by default, causes the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting to be ignored during bursts of I/O activity that occur at [checkpoints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checkpoint). To adhere to the I/O rate defined by the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting, disable [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync).

For information about configuring the [**innodb\_flush\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_sync) variable, see [Section 15.8.7, “Configuring InnoDB I/O Capacity”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-configuring-io-capacity).

**[innodb\_flushing\_avg\_loops](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_flushing_avg_loops)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-flushing-avg-loops=#** |
| **System Variable** | [**innodb\_flushing\_avg\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flushing_avg_loops) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **30** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1000** |

Number of iterations for which **InnoDB** keeps the previously calculated snapshot of the flushing state, controlling how quickly [adaptive flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_adaptive_flushing) responds to changing [workloads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload). Increasing the value makes the rate of [flush](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) operations change smoothly and gradually as the workload changes. Decreasing the value makes adaptive flushing adjust quickly to workload changes, which can cause spikes in flushing activity if the workload increases and decreases suddenly.

For related information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

**[innodb\_force\_load\_corrupted](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_force_load_corrupted)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-force-load-corrupted[={OFF|ON}]** |
| **System Variable** | [**innodb\_force\_load\_corrupted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_load_corrupted) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Permits **InnoDB** to load tables at startup that are marked as corrupted. Use only during troubleshooting, to recover data that is otherwise inaccessible. When troubleshooting is complete, disable this setting and restart the server.

**[innodb\_force\_recovery](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_force_recovery)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-force-recovery=#** |
| **System Variable** | [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **6** |

The [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) mode, typically only changed in serious troubleshooting situations. Possible values are from 0 to 6. For the meanings of these values and important information about [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery), see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

**Warning**

Only set this variable to a value greater than 0 in an emergency situation so that you can start **InnoDB** and dump your tables. As a safety measure, **InnoDB** prevents [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations when [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) is greater than 0. An [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) setting of 4 or greater places **InnoDB** into read-only mode.

These restrictions may cause replication administration commands to fail with an error, as replication stores the replica status logs in **InnoDB** tables.

**[innodb\_fsync\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_fsync_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-fsync-threshold=#** |
| **Introduced** | 8.0.13 |
| **System Variable** | [**innodb\_fsync\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fsync_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2\*\*64-1** |

By default, when **InnoDB** creates a new data file, such as a new log file or tablespace file, the file is fully written to the operating system cache before it is flushed to disk, which can cause a large amount of disk write activity to occur at once. To force smaller, periodic flushes of data from the operating system cache, you can use the [**innodb\_fsync\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fsync_threshold) variable to define a threshold value, in bytes. When the byte threshold is reached, the contents of the operating system cache are flushed to disk. The default value of 0 forces the default behavior, which is to flush data to disk only after a file is fully written to the cache.

Specifying a threshold to force smaller, periodic flushes may be beneficial in cases where multiple MySQL instances use the same storage devices. For example, creating a new MySQL instance and its associated data files could cause large surges of disk write activity, impeding the performance of other MySQL instances that use the same storage devices. Configuring a threshold helps avoid such surges in write activity.

**[innodb\_ft\_aux\_table](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_aux_table)**

|  |  |
| --- | --- |
| **System Variable** | [**innodb\_ft\_aux\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_aux_table) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

Specifies the qualified name of an **InnoDB** table containing a **FULLTEXT** index. This variable is intended for diagnostic purposes and can only be set at runtime. For example:

SET GLOBAL innodb\_ft\_aux\_table = 'test/t1';

After you set this variable to a name in the format ***db\_name*/*table\_name***, the **INFORMATION\_SCHEMA** tables [**INNODB\_FT\_INDEX\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-table-table), [**INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table), [**INNODB\_FT\_CONFIG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-config-table), [**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table), and [**INNODB\_FT\_BEING\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-being-deleted-table) show information about the search index for the specified table.

For more information, see [Section 15.15.4, “InnoDB INFORMATION\_SCHEMA FULLTEXT Index Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-fulltext_index-tables).

**[innodb\_ft\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-cache-size=#** |
| **System Variable** | [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **8000000** |
| **Minimum Value** | **1600000** |
| **Maximum Value** | **80000000** |

The memory allocated, in bytes, for the **InnoDB** **FULLTEXT** search index cache, which holds a parsed document in memory while creating an **InnoDB** **FULLTEXT** index. Index inserts and updates are only committed to disk when the [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) size limit is reached. [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) defines the cache size on a per table basis. To set a global limit for all tables, see [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size).

For more information, see [InnoDB Full-Text Index Cache](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-cache).

**[innodb\_ft\_enable\_diag\_print](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_enable_diag_print)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-enable-diag-print[={OFF|ON}]** |
| **System Variable** | [**innodb\_ft\_enable\_diag\_print**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_diag_print) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Whether to enable additional full-text search (FTS) diagnostic output. This option is primarily intended for advanced FTS debugging and is not of interest to most users. Output is printed to the error log and includes information such as:

FTS index sync progress (when the FTS cache limit is reached). For example:

FTS SYNC for table test, deleted count: 100 size: 10000 bytes

SYNC words: 100

FTS optimize progress. For example:

FTS start optimize test

FTS\_OPTIMIZE: optimize "mysql"

FTS\_OPTIMIZE: processed "mysql"

FTS index build progress. For example:

Number of doc processed: 1000

For FTS queries, the query parsing tree, word weight, query processing time, and memory usage are printed. For example:

FTS Search Processing time: 1 secs: 100 millisec: row(s) 10000

Full Search Memory: 245666 (bytes), Row: 10000

**[innodb\_ft\_enable\_stopword](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_enable_stopword)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-enable-stopword[={OFF|ON}]** |
| **System Variable** | [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies that a set of [stopwords](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_stopword) is associated with an **InnoDB** **FULLTEXT** index at the time the index is created. If the [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) option is set, the stopwords are taken from that table. Else, if the [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) option is set, the stopwords are taken from that table. Otherwise, a built-in set of default stopwords is used.

For more information, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

**[innodb\_ft\_max\_token\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_max_token_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-max-token-size=#** |
| **System Variable** | [**innodb\_ft\_max\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_max_token_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **84** |
| **Minimum Value** | **10** |
| **Maximum Value** | **84** |

Maximum character length of words that are stored in an **InnoDB** **FULLTEXT** index. Setting a limit on this value reduces the size of the index, thus speeding up queries, by omitting long keywords or arbitrary collections of letters that are not real words and are not likely to be search terms.

For more information, see [Section 12.10.6, “Fine-Tuning MySQL Full-Text Search”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning).

**[innodb\_ft\_min\_token\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_min_token_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-min-token-size=#** |
| **System Variable** | [**innodb\_ft\_min\_token\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_min_token_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **3** |
| **Minimum Value** | **0** |
| **Maximum Value** | **16** |

Minimum length of words that are stored in an **InnoDB** **FULLTEXT** index. Increasing this value reduces the size of the index, thus speeding up queries, by omitting common words that are unlikely to be significant in a search context, such as the English words “a” and “to”. For content using a CJK (Chinese, Japanese, Korean) character set, specify a value of 1.

For more information, see [Section 12.10.6, “Fine-Tuning MySQL Full-Text Search”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning).

**[innodb\_ft\_num\_word\_optimize](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_num_word_optimize)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-num-word-optimize=#** |
| **System Variable** | [**innodb\_ft\_num\_word\_optimize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_num_word_optimize) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2000** |
| **Minimum Value** | **1000** |
| **Maximum Value** | **10000** |

Number of words to process during each [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operation on an **InnoDB** **FULLTEXT** index. Because a bulk insert or update operation to a table containing a full-text search index could require substantial index maintenance to incorporate all changes, you might do a series of [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statements, each picking up where the last left off.

For more information, see [Section 12.10.6, “Fine-Tuning MySQL Full-Text Search”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning).

**[innodb\_ft\_result\_cache\_limit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_result_cache_limit)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-result-cache-limit=#** |
| **System Variable** | [**innodb\_ft\_result\_cache\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_result_cache_limit) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2000000000** |
| **Minimum Value** | **1000000** |
| **Maximum Value** | **2\*\*32-1** |

The **InnoDB** full-text search query result cache limit (defined in bytes) per full-text search query or per thread. Intermediate and final **InnoDB** full-text search query results are handled in memory. Use [**innodb\_ft\_result\_cache\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_result_cache_limit) to place a size limit on the full-text search query result cache to avoid excessive memory consumption in case of very large **InnoDB** full-text search query results (millions or hundreds of millions of rows, for example). Memory is allocated as required when a full-text search query is processed. If the result cache size limit is reached, an error is returned indicating that the query exceeds the maximum allowed memory.

The maximum value of [**innodb\_ft\_result\_cache\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_result_cache_limit) for all platform types and bit sizes is 2\*\*32-1.

**[innodb\_ft\_server\_stopword\_table](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_server_stopword_table)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-server-stopword-table=db\_name/table\_name** |
| **System Variable** | [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **NULL** |

This option is used to specify your own **InnoDB** **FULLTEXT** index stopword list for all **InnoDB** tables. To configure your own stopword list for a specific **InnoDB** table, use [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table).

Set [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) to the name of the table containing a list of stopwords, in the format ***db\_name*/*table\_name***.

The stopword table must exist before you configure [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table). [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword) must be enabled and [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table) option must be configured before you create the **FULLTEXT** index.

The stopword table must be an **InnoDB** table, containing a single **VARCHAR** column named **value**.

For more information, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

**[innodb\_ft\_sort\_pll\_degree](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_sort_pll_degree)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-sort-pll-degree=#** |
| **System Variable** | [**innodb\_ft\_sort\_pll\_degree**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_sort_pll_degree) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2** |
| **Minimum Value** | **1** |
| **Maximum Value** | **32** |

Number of threads used in parallel to index and tokenize text in an **InnoDB** **FULLTEXT** index when building a [search index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_search_index).

For related information, see [Section 15.6.2.4, “InnoDB Full-Text Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index), and [**innodb\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sort_buffer_size).

**[innodb\_ft\_total\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_total_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-total-cache-size=#** |
| **System Variable** | [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **640000000** |
| **Minimum Value** | **32000000** |
| **Maximum Value** | **1600000000** |

The total memory allocated, in bytes, for the **InnoDB** full-text search index cache for all tables. Creating numerous tables, each with a **FULLTEXT** search index, could consume a significant portion of available memory. [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size) defines a global memory limit for all full-text search indexes to help avoid excessive memory consumption. If the global limit is reached by an index operation, a forced sync is triggered.

For more information, see [InnoDB Full-Text Index Cache](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-fulltext-index-cache).

**[innodb\_ft\_user\_stopword\_table](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_ft_user_stopword_table)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-ft-user-stopword-table=db\_name/table\_name** |
| **System Variable** | [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **NULL** |

This option is used to specify your own **InnoDB** **FULLTEXT** index stopword list on a specific table. To configure your own stopword list for all **InnoDB** tables, use [**innodb\_ft\_server\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_server_stopword_table).

Set [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) to the name of the table containing a list of stopwords, in the format ***db\_name*/*table\_name***.

The stopword table must exist before you configure [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table). [**innodb\_ft\_enable\_stopword**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_enable_stopword) must be enabled and [**innodb\_ft\_user\_stopword\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_user_stopword_table) must be configured before you create the **FULLTEXT** index.

The stopword table must be an **InnoDB** table, containing a single **VARCHAR** column named **value**.

For more information, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

**[innodb\_idle\_flush\_pct](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_idle_flush_pct)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-idle-flush-pct=#** |
| **Introduced** | 8.0.18 |
| **System Variable** | [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **100** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |

Limits page flushing when **InnoDB** is idle. The [**innodb\_idle\_flush\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_idle_flush_pct) value is a percentage of the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) setting, which defines the number of I/O operations per second available to **InnoDB**. For more information, see [Limiting Buffer Flushing During Idle Periods](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-limit-flushing-rate).

**[innodb\_io\_capacity](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_io_capacity)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-io-capacity=#** |
| **System Variable** | [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **200** |
| **Minimum Value** | **100** |
| **Maximum Value (64-bit platforms)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms)** | **2\*\*32-1** |

The [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) variable defines the number of I/O operations per second (IOPS) available to **InnoDB** background tasks, such as [flushing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) pages from the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) and merging data from the [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer).

For information about configuring the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) variable, see [Section 15.8.7, “Configuring InnoDB I/O Capacity”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-configuring-io-capacity).

**[innodb\_io\_capacity\_max](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_io_capacity_max)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-io-capacity-max=#** |
| **System Variable** | [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **see description** |
| **Minimum Value** | **100** |
| **Maximum Value (Windows, 64-bit platforms)** | **2\*\*32-1** |
| **Maximum Value (Unix, 64-bit platforms)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms)** | **2\*\*32-1** |

If flushing activity falls behind, **InnoDB** can flush more aggressively, at a higher rate of I/O operations per second (IOPS) than defined by the [**innodb\_io\_capacity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity) variable. The [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) variable defines a maximum number of IOPS performed by **InnoDB** background tasks in such situations.

For information about configuring the [**innodb\_io\_capacity\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_io_capacity_max) variable, see [Section 15.8.7, “Configuring InnoDB I/O Capacity”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-configuring-io-capacity).

**[innodb\_limit\_optimistic\_insert\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_limit_optimistic_insert_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-limit-optimistic-insert-debug=#** |
| **System Variable** | [**innodb\_limit\_optimistic\_insert\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_limit_optimistic_insert_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2\*\*32-1** |

Limits the number of records per [B-tree](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) page. A default value of 0 means that no limit is imposed. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_lock\_wait\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_lock_wait_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-lock-wait-timeout=#** |
| **System Variable** | [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1073741824** |

The length of time in seconds an **InnoDB** [transaction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction) waits for a [row lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_row_lock) before giving up. The default value is 50 seconds. A transaction that tries to access a row that is locked by another **InnoDB** transaction waits at most this many seconds for write access to the row before issuing the following error:

ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction

When a lock wait timeout occurs, the current statement is [rolled back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback) (not the entire transaction). To have the entire transaction roll back, start the server with the [--innodb-rollback-on-timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) option. See also [Section 15.21.4, “InnoDB Error Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-error-handling).

You might decrease this value for highly interactive applications or [OLTP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_oltp) systems, to display user feedback quickly or put the update into a queue for processing later. You might increase this value for long-running back-end operations, such as a transform step in a data warehouse that waits for other large insert or update operations to finish.

[**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) applies to **InnoDB** row locks. A MySQL [table lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_table_lock) does not happen inside **InnoDB** and this timeout does not apply to waits for table locks.

The lock wait timeout value does not apply to [deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) when [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) is enabled (the default) because **InnoDB** detects deadlocks immediately and rolls back one of the deadlocked transactions. When [**innodb\_deadlock\_detect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_deadlock_detect) is disabled, **InnoDB** relies on [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) for transaction rollback when a deadlock occurs. See [Section 15.7.5.2, “Deadlock Detection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlock-detection).

[**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) can be set at runtime with the **SET GLOBAL** or **SET SESSION** statement. Changing the **GLOBAL** setting requires privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)) and affects the operation of all clients that subsequently connect. Any client can change the **SESSION** setting for [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout), which affects only that client.

**[innodb\_log\_buffer\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_buffer_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-buffer-size=#** |
| **System Variable** | [**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **16777216** |
| **Minimum Value** | **1048576** |
| **Maximum Value** | **4294967295** |

The size in bytes of the buffer that **InnoDB** uses to write to the [log files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file) on disk. The default is 16MB. A large [log buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_buffer) enables large [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction) to run without the need to write the log to disk before the transactions [commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit). Thus, if you have transactions that update, insert, or delete many rows, making the log buffer larger saves disk I/O. For related information, see [Memory Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-memory-configuration), and [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_log\_checkpoint\_fuzzy\_now](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_checkpoint_fuzzy_now)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-checkpoint-fuzzy-now[={OFF|ON}]** |
| **Introduced** | 8.0.13 |
| **System Variable** | [**innodb\_log\_checkpoint\_fuzzy\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checkpoint_fuzzy_now) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enable this debug option to force **InnoDB** to write a fuzzy checkpoint. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_log\_checkpoint\_now](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_checkpoint_now)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-checkpoint-now[={OFF|ON}]** |
| **System Variable** | [**innodb\_log\_checkpoint\_now**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checkpoint_now) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enable this debug option to force **InnoDB** to write a checkpoint. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_log\_checksums](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_checksums)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-checksums[={OFF|ON}]** |
| **System Variable** | [**innodb\_log\_checksums**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checksums) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Enables or disables checksums for redo log pages.

[**innodb\_log\_checksums=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checksums) enables the **CRC-32C** checksum algorithm for redo log pages. When [**innodb\_log\_checksums**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_checksums) is disabled, the contents of the redo log page checksum field are ignored.

Checksums on the redo log header page and redo log checkpoint pages are never disabled.

**[innodb\_log\_compressed\_pages](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_compressed_pages)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-compressed-pages[={OFF|ON}]** |
| **System Variable** | [**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies whether images of [re-compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) are written to the [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log). Re-compression may occur when changes are made to compressed data.

[**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) is enabled by default to prevent corruption that could occur if a different version of the **zlib** compression algorithm is used during recovery. If you are certain that the **zlib** version is not subject to change, you can disable [**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages) to reduce redo log generation for workloads that modify compressed data.

To measure the effect of enabling or disabling [**innodb\_log\_compressed\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_compressed_pages), compare redo log generation for both settings under the same workload. Options for measuring redo log generation include observing the **Log sequence number** (LSN) in the **LOG** section of [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output, or monitoring [**Innodb\_os\_log\_written**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_os_log_written) status for the number of bytes written to the redo log files.

For related information, see [Section 15.9.1.6, “Compression for OLTP Workloads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-compression-oltp).

**[innodb\_log\_file\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_file_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-file-size=#** |
| **System Variable** | [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50331648** |
| **Minimum Value** | **4194304** |
| **Maximum Value** | **512GB / innodb\_log\_files\_in\_group** |

The size in bytes of each [log file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file) in a [log group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_group). The combined size of log files ([**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) \* [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group)) cannot exceed a maximum value that is slightly less than 512GB. A pair of 255 GB log files, for example, approaches the limit but does not exceed it. The default value is 48MB.

Generally, the combined size of the log files should be large enough that the server can smooth out peaks and troughs in workload activity, which often means that there is enough redo log space to handle more than an hour of write activity. The larger the value, the less checkpoint flush activity is required in the buffer pool, saving disk I/O. Larger log files also make [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) slower.

The minimum [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) is 4MB.

For related information, see [Redo Log File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-log-file-configuration). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

If [**innodb\_dedicated\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_dedicated_server) is enabled, the [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) value is automatically configured if it is not explicitly defined. For more information, see [Section 15.8.12, “Enabling Automatic Configuration for a Dedicated MySQL Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-dedicated-server).

**[innodb\_log\_files\_in\_group](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_files_in_group)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-files-in-group=#** |
| **System Variable** | [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2** |
| **Minimum Value** | **2** |
| **Maximum Value** | **100** |

The number of [log files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_file) in the [log group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_log_group). **InnoDB** writes to the files in a circular fashion. The default (and recommended) value is 2. The location of the files is specified by [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir). The combined size of log files ([**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) \* [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group)) can be up to 512GB.

For related information, see [Redo Log File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-log-file-configuration).

**[innodb\_log\_group\_home\_dir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_group_home_dir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-group-home-dir=dir\_name** |
| **System Variable** | [**innodb\_log\_group\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_group_home_dir) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |

The directory path to the **InnoDB** [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) files, whose number is specified by [**innodb\_log\_files\_in\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_files_in_group). If you do not specify any **InnoDB** log variables, the default is to create two files named ib\_logfile0 and ib\_logfile1 in the MySQL data directory. Log file size is given by the [**innodb\_log\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_file_size) system variable.

For related information, see [Redo Log File Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-startup-log-file-configuration).

**[innodb\_log\_spin\_cpu\_abs\_lwm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_spin_cpu_abs_lwm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-spin-cpu-abs-lwm=#** |
| **System Variable** | [**innodb\_log\_spin\_cpu\_abs\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_spin_cpu_abs_lwm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **80** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

Defines the minimum amount of CPU usage below which user threads no longer spin while waiting for flushed redo. The value is expressed as a sum of CPU core usage. For example, The default value of 80 is 80% of a single CPU core. On a system with a multi-core processor, a value of 150 represents 100% usage of one CPU core plus 50% usage of a second CPU core.

For related information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**[innodb\_log\_spin\_cpu\_pct\_hwm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_spin_cpu_pct_hwm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-spin-cpu-pct-hwm=#** |
| **System Variable** | [**innodb\_log\_spin\_cpu\_pct\_hwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_spin_cpu_pct_hwm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100** |

Defines the maximum amount of CPU usage above which user threads no longer spin while waiting for flushed redo. The value is expressed as a percentage of the combined total processing power of all CPU cores. The default value is 50%. For example, 100% usage of two CPU cores is 50% of the combined CPU processing power on a server with four CPU cores.

The [**innodb\_log\_spin\_cpu\_pct\_hwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_spin_cpu_pct_hwm) variable respects processor affinity. For example, if a server has 48 cores but the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process is pinned to only four CPU cores, the other 44 CPU cores are ignored.

For related information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**[innodb\_log\_wait\_for\_flush\_spin\_hwm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_wait_for_flush_spin_hwm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-wait-for-flush-spin-hwm=#** |
| **System Variable** | [**innodb\_log\_wait\_for\_flush\_spin\_hwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_wait_for_flush_spin_hwm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **400** |
| **Minimum Value** | **0** |
| **Maximum Value (64-bit platforms)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms)** | **2\*\*32-1** |

Defines the maximum average log flush time beyond which user threads no longer spin while waiting for flushed redo. The default value is 400 microseconds.

For related information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**[innodb\_log\_write\_ahead\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_write_ahead_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-write-ahead-size=#** |
| **System Variable** | [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **8192** |
| **Minimum Value** | **512 (log file block size)** |
| **Maximum Value** | **Equal to innodb\_page\_size** |

Defines the write-ahead block size for the redo log, in bytes. To avoid “read-on-write”, set [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) to match the operating system or file system cache block size. The default setting is 8192 bytes. Read-on-write occurs when redo log blocks are not entirely cached to the operating system or file system due to a mismatch between write-ahead block size for the redo log and operating system or file system cache block size.

Valid values for [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) are multiples of the **InnoDB** log file block size (2n). The minimum value is the **InnoDB** log file block size (512). Write-ahead does not occur when the minimum value is specified. The maximum value is equal to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. If you specify a value for [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) that is larger than the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value, the [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) setting is truncated to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value.

Setting the [**innodb\_log\_write\_ahead\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_write_ahead_size) value too low in relation to the operating system or file system cache block size results in “read-on-write”. Setting the value too high may have a slight impact on **fsync** performance for log file writes due to several blocks being written at once.

For related information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**[innodb\_log\_writer\_threads](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_log_writer_threads)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-log-writer-threads[={OFF|ON}]** |
| **Introduced** | 8.0.22 |
| **System Variable** | [**innodb\_log\_writer\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_writer_threads) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Enables dedicated log writer threads for writing redo log records from the log buffer to the system buffers and flushing the system buffers to the redo log files. Dedicated log writer threads can improve performance on high-concurrency systems, but for low-concurrency systems, disabling dedicated log writer threads provides better performance.

For more information, see [Section 8.5.4, “Optimizing InnoDB Redo Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-logging).

**[innodb\_lru\_scan\_depth](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_lru_scan_depth)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-lru-scan-depth=#** |
| **System Variable** | [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1024** |
| **Minimum Value** | **100** |
| **Maximum Value (64-bit platforms)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms)** | **2\*\*32-1** |

A parameter that influences the algorithms and heuristics for the [flush](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) operation for the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). Primarily of interest to performance experts tuning I/O-intensive workloads. It specifies, per buffer pool instance, how far down the buffer pool LRU page list the page cleaner thread scans looking for [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) to flush. This is a background operation performed once per second.

A setting smaller than the default is generally suitable for most workloads. A value that is much higher than necessary may impact performance. Only consider increasing the value if you have spare I/O capacity under a typical workload. Conversely, if a write-intensive workload saturates your I/O capacity, decrease the value, especially in the case of a large buffer pool.

When tuning [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth), start with a low value and configure the setting upward with the goal of rarely seeing zero free pages. Also, consider adjusting [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) when changing the number of buffer pool instances, since [**innodb\_lru\_scan\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lru_scan_depth) \* [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) defines the amount of work performed by the page cleaner thread each second.

For related information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_max\_dirty\_pages\_pct](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_max_dirty_pages_pct)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-max-dirty-pages-pct=#** |
| **System Variable** | [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Numeric |
| **Default Value** | **90** |
| **Minimum Value** | **0** |
| **Maximum Value** | **99.99** |

**InnoDB** tries to [flush](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_flush) data from the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) so that the percentage of [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) does not exceed this value.

The [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) setting establishes a target for flushing activity. It does not affect the rate of flushing. For information about managing the rate of flushing, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

For related information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_max\_dirty\_pages\_pct\_lwm](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_max_dirty_pages_pct_lwm)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-max-dirty-pages-pct-lwm=#** |
| **System Variable** | [**innodb\_max\_dirty\_pages\_pct\_lwm**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct_lwm) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Numeric |
| **Default Value** | **10** |
| **Minimum Value** | **0** |
| **Maximum Value** | **99.99** |

Defines a low water mark representing the percentage of [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) at which preflushing is enabled to control the dirty page ratio. A value of 0 disables the pre-flushing behavior entirely. The configured value should always be lower than the [**innodb\_max\_dirty\_pages\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_dirty_pages_pct) value. For more information, see [Section 15.8.3.5, “Configuring Buffer Pool Flushing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-flushing).

**[innodb\_max\_purge\_lag](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_max_purge_lag)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-max-purge-lag=#** |
| **System Variable** | [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

Defines the desired maximum purge lag. If this value is exceeded, a delay is imposed on [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations to allow time for purge to catch up. The default value is 0, which means there is no maximum purge lag and no delay.

For more information, see [Section 15.8.9, “Purge Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration).

**[innodb\_max\_purge\_lag\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_max_purge_lag_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-max-purge-lag-delay=#** |
| **System Variable** | [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **10000000** |

Specifies the maximum delay in microseconds for the delay imposed when the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) threshold is exceeded. The specified [**innodb\_max\_purge\_lag\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag_delay) value is an upper limit on the delay period calculated by the [**innodb\_max\_purge\_lag**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_purge_lag) formula.

For more information, see [Section 15.8.9, “Purge Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration).

**[innodb\_max\_undo\_log\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_max_undo_log_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-max-undo-log-size=#** |
| **System Variable** | [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1073741824** |
| **Minimum Value** | **10485760** |
| **Maximum Value** | **2\*\*64-1** |

Defines a threshold size for undo tablespaces. If an undo tablespace exceeds the threshold, it can be marked for truncation when [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) is enabled. The default value is 1073741824 bytes (1024 MiB).

For more information, see [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

**[innodb\_merge\_threshold\_set\_all\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_merge_threshold_set_all_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-merge-threshold-set-all-debug=#** |
| **System Variable** | [**innodb\_merge\_threshold\_set\_all\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_merge_threshold_set_all_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **1** |
| **Maximum Value** | **50** |

Defines a page-full percentage value for index pages that overrides the current **MERGE\_THRESHOLD** setting for all indexes that are currently in the dictionary cache. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option. For related information, see [Section 15.8.11, “Configuring the Merge Threshold for Index Pages”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold).

**[innodb\_monitor\_disable](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_monitor_disable)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-monitor-disable={counter|module|pattern|all}** |
| **System Variable** | [**innodb\_monitor\_disable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

Disables **InnoDB** [metrics counters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metrics_counter). Counter data may be queried using the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table. For usage information, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

[**innodb\_monitor\_disable='latch'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable) disables statistics collection for [**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine). For more information, see [Section 13.7.7.15, “SHOW ENGINE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine).

**[innodb\_monitor\_enable](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_monitor_enable)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-monitor-enable={counter|module|pattern|all}** |
| **System Variable** | [**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

Enables **InnoDB** [metrics counters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metrics_counter). Counter data may be queried using the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table. For usage information, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

[**innodb\_monitor\_enable='latch'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) enables statistics collection for [**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine). For more information, see [Section 13.7.7.15, “SHOW ENGINE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine).

**[innodb\_monitor\_reset](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_monitor_reset)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-monitor-reset={counter|module|pattern|all}** |
| **System Variable** | [**innodb\_monitor\_reset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **empty string** |
| **Valid Values** | **counter**  **module**  **pattern**  **all** |

Resets the count value for **InnoDB** [metrics counters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metrics_counter) to zero. Counter data may be queried using the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table. For usage information, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

[**innodb\_monitor\_reset='latch'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset) resets statistics reported by [**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine). For more information, see [Section 13.7.7.15, “SHOW ENGINE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine).

**[innodb\_monitor\_reset\_all](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_monitor_reset_all)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-monitor-reset-all={counter|module|pattern|all}** |
| **System Variable** | [**innodb\_monitor\_reset\_all**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset_all) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **empty string** |
| **Valid Values** | **counter**  **module**  **pattern**  **all** |

Resets all values (minimum, maximum, and so on) for **InnoDB** [metrics counters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metrics_counter). Counter data may be queried using the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table. For usage information, see [Section 15.15.6, “InnoDB INFORMATION\_SCHEMA Metrics Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table).

**[innodb\_numa\_interleave](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_numa_interleave)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-numa-interleave[={OFF|ON}]** |
| **System Variable** | [**innodb\_numa\_interleave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_numa_interleave) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables the NUMA interleave memory policy for allocation of the **InnoDB** buffer pool. When [**innodb\_numa\_interleave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_numa_interleave) is enabled, the NUMA memory policy is set to **MPOL\_INTERLEAVE** for the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process. After the **InnoDB** buffer pool is allocated, the NUMA memory policy is set back to **MPOL\_DEFAULT**. For the [**innodb\_numa\_interleave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_numa_interleave) option to be available, MySQL must be compiled on a NUMA-enabled Linux system.

**CMake** sets the default [WITH\_NUMA](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_numa) value based on whether the current platform has **NUMA** support. For more information, see [Section 2.9.7, “MySQL Source-Configuration Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#source-configuration-options).

**[innodb\_old\_blocks\_pct](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_old_blocks_pct)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-old-blocks-pct=#** |
| **System Variable** | [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **37** |
| **Minimum Value** | **5** |
| **Maximum Value** | **95** |

Specifies the approximate percentage of the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) used for the old block [sublist](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_sublist). The range of values is 5 to 95. The default value is 37 (that is, 3/8 of the pool). Often used in combination with [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time).

For more information, see [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion). For information about buffer pool management, the [LRU](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lru) algorithm, and [eviction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction) policies, see [Section 15.5.1, “Buffer Pool”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool).

**[innodb\_old\_blocks\_time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_old_blocks_time)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-old-blocks-time=#** |
| **System Variable** | [**innodb\_old\_blocks\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_time) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **2\*\*32-1** |

Non-zero values protect against the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) being filled by data that is referenced only for a brief period, such as during a [full table scan](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_full_table_scan). Increasing this value offers more protection against full table scans interfering with data cached in the buffer pool.

Specifies how long in milliseconds a block inserted into the old [sublist](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_sublist) must stay there after its first access before it can be moved to the new sublist. If the value is 0, a block inserted into the old sublist moves immediately to the new sublist the first time it is accessed, no matter how soon after insertion the access occurs. If the value is greater than 0, blocks remain in the old sublist until an access occurs at least that many milliseconds after the first access. For example, a value of 1000 causes blocks to stay in the old sublist for 1 second after the first access before they become eligible to move to the new sublist.

The default value is 1000.

This variable is often used in combination with [**innodb\_old\_blocks\_pct**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_old_blocks_pct). For more information, see [Section 15.8.3.3, “Making the Buffer Pool Scan Resistant”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-midpoint_insertion). For information about buffer pool management, the [LRU](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_lru) algorithm, and [eviction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction) policies, see [Section 15.5.1, “Buffer Pool”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool).

**[innodb\_online\_alter\_log\_max\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_online_alter_log_max_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-online-alter-log-max-size=#** |
| **System Variable** | [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **134217728** |
| **Minimum Value** | **65536** |
| **Maximum Value** | **2\*\*64-1** |

Specifies an upper limit in bytes on the size of the temporary log files used during [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_online_ddl) operations for **InnoDB** tables. There is one such log file for each index being created or table being altered. This log file stores data inserted, updated, or deleted in the table during the DDL operation. The temporary log file is extended when needed by the value of [**innodb\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sort_buffer_size), up to the maximum specified by [**innodb\_online\_alter\_log\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_online_alter_log_max_size). If a temporary log file exceeds the upper size limit, the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation fails and all uncommitted concurrent DML operations are rolled back. Thus, a large value for this option allows more DML to happen during an online DDL operation, but also extends the period of time at the end of the DDL operation when the table is locked to apply the data from the log.

**[innodb\_open\_files](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_open_files)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-open-files=#** |
| **System Variable** | [**innodb\_open\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_open_files) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **-1** (signifies autosizing; do not assign this literal value) |
| **Minimum Value** | **10** |
| **Maximum Value** | **4294967295** |

This variable is only relevant if you have numerous **InnoDB** [tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_tablespace). It specifies the maximum number of [.ibd files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) that MySQL can keep open at one time. The minimum value is 10. The default value is 300 if [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is not enabled, and the higher of 300 and [**table\_open\_cache**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_open_cache) otherwise.

The file descriptors used for .ibd files are for **InnoDB** tables only. They are independent of those specified by the [**open\_files\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_open_files_limit) system variable, and do not affect the operation of the table cache. For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

Prior to MySQL 8.0.24, temporary tablespaces were counted as open files, which could cause the [**innodb\_open\_files**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_open_files) limit to be exceeded, preventing other files from being opened. As of MySQL 8.0.24, temporary tablespaces are not counted as open files.

**[innodb\_optimize\_fulltext\_only](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_optimize_fulltext_only)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-optimize-fulltext-only[={OFF|ON}]** |
| **System Variable** | [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Changes the way [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operates on **InnoDB** tables. Intended to be enabled temporarily, during maintenance operations for **InnoDB** tables with **FULLTEXT** indexes.

By default, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) reorganizes data in the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index) of the table. When this option is enabled, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) skips the reorganization of table data, and instead processes newly added, deleted, and updated token data for **InnoDB** **FULLTEXT** indexes. For more information, see [Optimizing InnoDB Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-optimize).

**[innodb\_page\_cleaners](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_page_cleaners)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-page-cleaners=#** |
| **System Variable** | [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **4** |
| **Minimum Value** | **1** |
| **Maximum Value** | **64** |

The number of page cleaner threads that flush dirty pages from buffer pool instances. Page cleaner threads perform flush list and LRU flushing. When there are multiple page cleaner threads, buffer pool flushing tasks for each buffer pool instance are dispatched to idle page cleaner threads. The [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) default value is 4. If the number of page cleaner threads exceeds the number of buffer pool instances, [**innodb\_page\_cleaners**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_cleaners) is automatically set to the same value as [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances).

If your workload is write-IO bound when flushing dirty pages from buffer pool instances to data files, and if your system hardware has available capacity, increasing the number of page cleaner threads may help improve write-IO throughput.

Multithreaded page cleaner support extends to shutdown and recovery phases.

The **setpriority()** system call is used on Linux platforms where it is supported, and where the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) execution user is authorized to give **page\_cleaner** threads priority over other MySQL and **InnoDB** threads to help page flushing keep pace with the current workload. **setpriority()** support is indicated by this **InnoDB** startup message:

[Note] InnoDB: If the mysqld execution user is authorized, page cleaner

thread priority can be changed. See the man page of setpriority().

For systems where server startup and shutdown is not managed by systemd, [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) execution user authorization can be configured in /etc/security/limits.conf. For example, if [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is run under the **mysql** user, you can authorize the **mysql** user by adding these lines to /etc/security/limits.conf:

mysql hard nice -20

mysql soft nice -20

For systemd managed systems, the same can be achieved by specifying **LimitNICE=-20** in a localized systemd configuration file. For example, create a file named override.conf in /etc/systemd/system/mysqld.service.d/override.conf and add this entry:

[Service]

LimitNICE=-20

After creating or changing override.conf, reload the systemd configuration, then tell systemd to restart the MySQL service:

systemctl daemon-reload

systemctl restart mysqld # RPM platforms

systemctl restart mysql # Debian platforms

For more information about using a localized systemd configuration file, see [Configuring systemd for MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#systemd-mysql-configuration).

After authorizing the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) execution user, use the **cat** command to verify the configured **Nice** limits for the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process:

shell> cat /proc/***mysqld\_pid***/limits | grep nice

Max nice priority 18446744073709551596 18446744073709551596

**[innodb\_page\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_page_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-page-size=#** |
| **System Variable** | [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **16384** |
| **Valid Values** | **4096**  **8192**  **16384**  **32768**  **65536** |

Specifies the [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size) for **InnoDB** [tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_tablespace). Values can be specified in bytes or kilobytes. For example, a 16 kilobyte page size value can be specified as 16384, 16KB, or 16k.

[**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) can only be configured prior to initializing the MySQL instance and cannot be changed afterward. If no value is specified, the instance is initialized using the default page size. See [Section 15.8.1, “InnoDB Startup Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-init-startup-configuration).

For both 32KB and 64KB page sizes, the maximum row length is approximately 16000 bytes. **ROW\_FORMAT=COMPRESSED** is not supported when [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) is set to 32KB or 64KB. For [**innodb\_page\_size=32KB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), extent size is 2MB. For [**innodb\_page\_size=64KB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), extent size is 4MB. [**innodb\_log\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_log_buffer_size) should be set to at least 16M (the default) when using 32KB or 64KB page sizes.

The default 16KB page size or larger is appropriate for a wide range of [workloads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload), particularly for queries involving table scans and DML operations involving bulk updates. Smaller page sizes might be more efficient for [OLTP](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_oltp) workloads involving many small writes, where contention can be an issue when single pages contain many rows. Smaller pages might also be efficient with [SSD](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ssd) storage devices, which typically use small block sizes. Keeping the **InnoDB** page size close to the storage device block size minimizes the amount of unchanged data that is rewritten to disk.

The minimum file size for the first system tablespace data file (**ibdata1**) differs depending on the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. See the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) option description for more information.

A MySQL instance using a particular **InnoDB** page size cannot use data files or log files from an instance that uses a different page size.

For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_parallel\_read\_threads](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_parallel_read_threads)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-parallel-read-threads=#** |
| **Introduced** | 8.0.14 |
| **System Variable** | [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **4** |
| **Minimum Value** | **1** |
| **Maximum Value** | **256** |

Defines the number of threads that can be used for parallel clustered index reads. Parallel scanning of partitions is supported as of MySQL 8.0.17. Parallel read threads can improve [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) performance. **InnoDB** reads the clustered index twice during a [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) operation. The second read can be performed in parallel. This feature does not apply to secondary index scans. The [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) session variable must be set to a value greater than 1 for parallel clustered index reads to occur. The actual number of threads used to perform a parallel clustered index read is determined by the [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) setting or the number of index subtrees to scan, whichever is smaller. The pages read into the buffer pool during the scan are kept at the tail of the buffer pool LRU list so that they can be discarded quickly when free buffer pool pages are required.

As of MySQL 8.0.17, the maximum number of parallel read threads (256) is the total number of threads for all client connections. If the thread limit is reached, connections fall back to using a single thread.

**[innodb\_print\_all\_deadlocks](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_print_all_deadlocks)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-print-all-deadlocks[={OFF|ON}]** |
| **System Variable** | [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

When this option is enabled, information about all [deadlocks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) in **InnoDB** user transactions is recorded in the **mysqld** [error log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log). Otherwise, you see information about only the last deadlock, using the **SHOW ENGINE INNODB STATUS** command. An occasional **InnoDB** deadlock is not necessarily an issue, because **InnoDB** detects the condition immediately and rolls back one of the transactions automatically. You might use this option to troubleshoot why deadlocks are occurring if an application does not have appropriate error-handling logic to detect the rollback and retry its operation. A large number of deadlocks might indicate the need to restructure transactions that issue [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) or **SELECT ... FOR UPDATE** statements for multiple tables, so that each transaction accesses the tables in the same order, thus avoiding the deadlock condition.

For related information, see [Section 15.7.5, “Deadlocks in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks).

**[innodb\_print\_ddl\_logs](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_print_ddl_logs)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-print-ddl-logs[={OFF|ON}]** |
| **System Variable** | [**innodb\_print\_ddl\_logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_ddl_logs) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enabling this option causes MySQL to write DDL logs to **stderr**. For more information, see [Viewing DDL Logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-view-logs).

**[innodb\_purge\_batch\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_purge_batch_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-purge-batch-size=#** |
| **System Variable** | [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **300** |
| **Minimum Value** | **1** |
| **Maximum Value** | **5000** |

Defines the number of undo log pages that purge parses and processes in one batch from the [history list](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_history_list). In a multithreaded purge configuration, the coordinator purge thread divides [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) by [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) and assigns that number of pages to each purge thread. The [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) variable also defines the number of undo log pages that purge frees after every 128 iterations through the undo logs.

The [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) option is intended for advanced performance tuning in combination with the [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) setting. Most users need not change [**innodb\_purge\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_batch_size) from its default value.

For related information, see [Section 15.8.9, “Purge Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration).

**[innodb\_purge\_threads](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_purge_threads)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-purge-threads=#** |
| **System Variable** | [**innodb\_purge\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_threads) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **4** |
| **Minimum Value** | **1** |
| **Maximum Value** | **32** |

The number of background threads devoted to the **InnoDB** [purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge) operation. Increasing the value creates additional purge threads, which can improve efficiency on systems where [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) operations are performed on multiple tables.

For related information, see [Section 15.8.9, “Purge Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-purge-configuration).

**[innodb\_purge\_rseg\_truncate\_frequency](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_purge_rseg_truncate_frequency)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-purge-rseg-truncate-frequency=#** |
| **System Variable** | [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **128** |
| **Minimum Value** | **1** |
| **Maximum Value** | **128** |

Defines the frequency with which the purge system frees rollback segments in terms of the number of times that purge is invoked. An undo tablespace cannot be truncated until its rollback segments are freed. Normally, the purge system frees rollback segments once every 128 times that purge is invoked. The default value is 128. Reducing this value increases the frequency with which the purge thread frees rollback segments.

[**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) is intended for use with [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate). For more information, see [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

**[innodb\_random\_read\_ahead](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_random_read_ahead)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-random-read-ahead[={OFF|ON}]** |
| **System Variable** | [**innodb\_random\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_random_read_ahead) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables the random [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) technique for optimizing **InnoDB** I/O.

For details about performance considerations for different types of read-ahead requests, see [Section 15.8.3.4, “Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_read\_ahead\_threshold](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_read_ahead_threshold)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-read-ahead-threshold=#** |
| **System Variable** | [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **56** |
| **Minimum Value** | **0** |
| **Maximum Value** | **64** |

Controls the sensitivity of linear [read-ahead](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_read_ahead) that **InnoDB** uses to prefetch pages into the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). If **InnoDB** reads at least [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) pages sequentially from an [extent](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_extent) (64 pages), it initiates an asynchronous read for the entire following extent. The permissible range of values is 0 to 64. A value of 0 disables read-ahead. For the default of 56, **InnoDB** must read at least 56 pages sequentially from an extent to initiate an asynchronous read for the following extent.

Knowing how many pages are read through the read-ahead mechanism, and how many of these pages are evicted from the buffer pool without ever being accessed, can be useful when fine-tuning the [**innodb\_read\_ahead\_threshold**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_ahead_threshold) setting. [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output displays counter information from the [**Innodb\_buffer\_pool\_read\_ahead**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead) and [**Innodb\_buffer\_pool\_read\_ahead\_evicted**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Innodb_buffer_pool_read_ahead_evicted) global status variables, which report the number of pages brought into the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) by read-ahead requests, and the number of such pages [evicted](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_eviction) from the buffer pool without ever being accessed, respectively. The status variables report global values since the last server restart.

[**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) also shows the rate at which the read-ahead pages are read and the rate at which such pages are evicted without being accessed. The per-second averages are based on the statistics collected since the last invocation of **SHOW ENGINE INNODB STATUS** and are displayed in the **BUFFER POOL AND MEMORY** section of the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output.

For more information, see [Section 15.8.3.4, “Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-read_ahead). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**[innodb\_read\_io\_threads](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_read_io_threads)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-read-io-threads=#** |
| **System Variable** | [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **4** |
| **Minimum Value** | **1** |
| **Maximum Value** | **64** |

The number of I/O threads for read operations in **InnoDB**. Its counterpart for write threads is [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads). For more information, see [Section 15.8.5, “Configuring the Number of Background InnoDB I/O Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-multiple_io_threads). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**Note**

On Linux systems, running multiple MySQL servers (typically more than 12) with default settings for [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads), [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads), and the Linux **aio-max-nr** setting can exceed system limits. Ideally, increase the **aio-max-nr** setting; as a workaround, you might reduce the settings for one or both of the MySQL variables.

**[innodb\_read\_only](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_read_only)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-read-only[={OFF|ON}]** |
| **System Variable** | [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Starts **InnoDB** in read-only mode. For distributing database applications or data sets on read-only media. Can also be used in data warehouses to share the same data directory between multiple instances. For more information, see [Section 15.8.2, “Configuring InnoDB for Read-Only Operation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-read-only-instance).

Previously, enabling the [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) system variable prevented creating and dropping tables only for the **InnoDB** storage engine. As of MySQL 8.0, enabling [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) prevents these operations for all storage engines. Table creation and drop operations for any storage engine modify data dictionary tables in the **mysql** system database, but those tables use the **InnoDB** storage engine and cannot be modified when [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) is enabled. The same principle applies to other table operations that require modifying data dictionary tables. Examples:

If the [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) system variable is enabled, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) may fail because it cannot update statistics tables in the data dictionary, which use **InnoDB**. For [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) operations that update the key distribution, failure may occur even if the operation updates the table itself (for example, if it is a **MyISAM** table). To obtain the updated distribution statistics, set [**information\_schema\_stats\_expiry=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_information_schema_stats_expiry).

[**ALTER TABLE *tbl\_name* ENGINE=*engine\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) fails because it updates the storage engine designation, which is stored in the data dictionary.

In addition, other tables in the **mysql** system database use the **InnoDB** storage engine in MySQL 8.0. Making those tables read only results in restrictions on operations that modify them. Examples:

Account-management statements such as [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) and [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) fail because the grant tables use **InnoDB**.

The [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) plugin-management statements fail because the **mysql.plugin** system table uses **InnoDB**.

The [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf) and [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) UDF-management statements fail because the **mysql.func** system table uses **InnoDB**.

**[innodb\_redo\_log\_archive\_dirs](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_redo_log_archive_dirs)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-redo-log-archive-dirs** |
| **Introduced** | 8.0.17 |
| **System Variable** | [**innodb\_redo\_log\_archive\_dirs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_archive_dirs) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **NULL** |

Defines labeled directories where redo log archive files can be created. You can define multiple labeled directories in a semicolon-separated list. For example:

innodb\_redo\_log\_archive\_dirs='label1:/backups1;label2:/backups2'

A label can be any string of characters, with the exception of colons (:), which are not permitted. An empty label is also permitted, but the colon (:) is still required in this case.

A path must be specified, and the directory must exist. The path can contain colons (':'), but semicolons (;) are not permitted.

**[innodb\_redo\_log\_encrypt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_redo_log_encrypt)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-redo-log-encrypt[={OFF|ON}]** |
| **System Variable** | [**innodb\_redo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_redo_log_encrypt) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Controls encryption of redo log data for tables encrypted using the **InnoDB** [data-at-rest encryption feature](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption). Encryption of redo log data is disabled by default. For more information, see [Redo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-redo-log).

**[innodb\_replication\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_replication_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-replication-delay=#** |
| **System Variable** | [**innodb\_replication\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_replication_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The replication thread delay in milliseconds on a replica server if [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is reached.

**[innodb\_rollback\_on\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_rollback_on_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-rollback-on-timeout[={OFF|ON}]** |
| **System Variable** | [**innodb\_rollback\_on\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

**InnoDB** [rolls back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback) only the last statement on a transaction timeout by default. If [--innodb-rollback-on-timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) is specified, a transaction timeout causes **InnoDB** to abort and roll back the entire transaction.

For more information, see [Section 15.21.4, “InnoDB Error Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-error-handling).

**[innodb\_rollback\_segments](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_rollback_segments)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-rollback-segments=#** |
| **System Variable** | [**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **128** |
| **Minimum Value** | **1** |
| **Maximum Value** | **128** |

[**innodb\_rollback\_segments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_segments) defines the number of [rollback segments](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback_segment) allocated to each undo tablespace and the global temporary tablespace for transactions that generate undo records. The number of transactions that each rollback segment supports depends on the **InnoDB** page size and the number of undo logs assigned to each transaction. For more information, see [Section 15.6.6, “Undo Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-logs).

For related information, see [Section 15.3, “InnoDB Multi-Versioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multi-versioning). For information about undo tablespaces, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

**[innodb\_saved\_page\_number\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_saved_page_number_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-saved-page-number-debug=#** |
| **System Variable** | [**innodb\_saved\_page\_number\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_saved_page_number_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Maximum Value** | **2\*\*23-1** |

Saves a page number. Setting the [**innodb\_fil\_make\_page\_dirty\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fil_make_page_dirty_debug) option dirties the page defined by [**innodb\_saved\_page\_number\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_saved_page_number_debug). The [**innodb\_saved\_page\_number\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_saved_page_number_debug) option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_segment\_reserve\_factor](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_segment_reserve_factor)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-segment-reserve-factor=#** |
| **Introduced** | 8.0.25 |
| **System Variable** | [**innodb\_segment\_reserve\_factor**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_segment_reserve_factor) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Numeric |
| **Default Value** | **12.5** |
| **Minimum Value** | **0.03** |
| **Maximum Value** | **40** |

Defines the percentage of tablespace file segment pages reserved as empty pages.

**[innodb\_sort\_buffer\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_sort_buffer_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-sort-buffer-size=#** |
| **System Variable** | [**innodb\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sort_buffer_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1048576** |
| **Minimum Value** | **65536** |
| **Maximum Value** | **67108864** |

Specifies the size of sort buffers used to sort data during creation of an **InnoDB** index. The specified size defines the amount of data that is read into memory for internal sorting and then written out to disk. This process is referred to as a “run”. During the merge phase, pairs of buffers of the specified size are read and merged. The larger the setting, the fewer runs and merges there are.

This sort area is only used for merge sorts during index creation, not during later index maintenance operations. Buffers are deallocated when index creation completes.

The value of this option also controls the amount by which the temporary log file is extended to record concurrent DML during [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_online_ddl) operations.

Before this setting was made configurable, the size was hardcoded to 1048576 bytes (1MB), which remains the default.

During an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement that creates an index, 3 buffers are allocated, each with a size defined by this option. Additionally, auxiliary pointers are allocated to rows in the sort buffer so that the sort can run on pointers (as opposed to moving rows during the sort operation).

For a typical sort operation, a formula such as this one can be used to estimate memory consumption:

(6 /\*FTS\_NUM\_AUX\_INDEX\*/ \* (3\*@@GLOBAL.innodb\_sort\_buffer\_size)

+ 2 \* number\_of\_partitions \* number\_of\_secondary\_indexes\_created

\* (@@GLOBAL.innodb\_sort\_buffer\_size/dict\_index\_get\_min\_size(index)\*/)

\* 8 /\*64-bit sizeof \*buf->tuples\*/")

**@@GLOBAL.innodb\_sort\_buffer\_size/dict\_index\_get\_min\_size(index)** indicates the maximum tuples held. **2 \* (@@GLOBAL.innodb\_sort\_buffer\_size/\*dict\_index\_get\_min\_size(index)\*/) \* 8 /\*64-bit size of \*buf->tuples\*/** indicates auxiliary pointers allocated.

**Note**

For 32-bit, multiply by 4 instead of 8.

For parallel sorts on a full-text index, multiply by the [**innodb\_ft\_sort\_pll\_degree**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_sort_pll_degree) setting:

(6 /\*FTS\_NUM\_AUX\_INDEX\*/ \* @@GLOBAL.innodb\_ft\_sort\_pll\_degree)

**[innodb\_spin\_wait\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_spin_wait_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-spin-wait-delay=#** |
| **System Variable** | [**innodb\_spin\_wait\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **6** |
| **Minimum Value** | **0** |
| **Maximum Value (64-bit platforms, ≤ 8.0.13)** | **2\*\*64-1** |
| **Maximum Value (32-bit platforms, ≤ 8.0.13)** | **2\*\*32-1** |
| **Maximum Value (≥ 8.0.14)** | **1000** |

The maximum delay between polls for a [spin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_spin) lock. The low-level implementation of this mechanism varies depending on the combination of hardware and operating system, so the delay does not correspond to a fixed time interval.

Can be used in combination with the [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) variable for greater control over the duration of spin-lock polling delays.

For more information, see [Section 15.8.8, “Configuring Spin Lock Polling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-spin_lock_polling).

**[innodb\_spin\_wait\_pause\_multiplier](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_spin_wait_pause_multiplier)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-spin-wait-pause-multiplier=#** |
| **Introduced** | 8.0.16 |
| **System Variable** | [**innodb\_spin\_wait\_pause\_multiplier**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_spin_wait_pause_multiplier) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **50** |
| **Minimum Value** | **1** |
| **Maximum Value** | **100** |

Defines a multiplier value used to determine the number of PAUSE instructions in spin-wait loops that occur when a thread waits to acquire a mutex or rw-lock.

For more information, see [Section 15.8.8, “Configuring Spin Lock Polling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-spin_lock_polling).

**[innodb\_stats\_auto\_recalc](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_auto_recalc)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-auto-recalc[={OFF|ON}]** |
| **System Variable** | [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Causes **InnoDB** to automatically recalculate [persistent statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_persistent_statistics) after the data in a table is changed substantially. The threshold value is 10% of the rows in the table. This setting applies to tables created when the [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) option is enabled. Automatic statistics recalculation may also be configured by specifying **STATS\_PERSISTENT=1** in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. The amount of data sampled to produce the statistics is controlled by the [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) variable.

For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**[innodb\_stats\_include\_delete\_marked](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_include_delete_marked)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-include-delete-marked[={OFF|ON}]** |
| **System Variable** | [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

By default, **InnoDB** reads uncommitted data when calculating statistics. In the case of an uncommitted transaction that deletes rows from a table, **InnoDB** excludes records that are delete-marked when calculating row estimates and index statistics, which can lead to non-optimal execution plans for other transactions that are operating on the table concurrently using a transaction isolation level other than [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted). To avoid this scenario, [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) can be enabled to ensure that **InnoDB** includes delete-marked records when calculating persistent optimizer statistics.

When [**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) is enabled, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) considers delete-marked records when recalculating statistics.

[**innodb\_stats\_include\_delete\_marked**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_include_delete_marked) is a global setting that affects all **InnoDB** tables. It is only applicable to persistent optimizer statistics.

For related information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**[innodb\_stats\_method](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_method)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-method=value** |
| **System Variable** | [**innodb\_stats\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_method) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **nulls\_equal** |
| **Valid Values** | **nulls\_equal**  **nulls\_unequal**  **nulls\_ignored** |

How the server treats **NULL** values when collecting [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) about the distribution of index values for **InnoDB** tables. Permitted values are **nulls\_equal**, **nulls\_unequal**, and **nulls\_ignored**. For **nulls\_equal**, all **NULL** index values are considered equal and form a single value group with a size equal to the number of **NULL** values. For **nulls\_unequal**, **NULL** values are considered unequal, and each **NULL** forms a distinct value group of size 1. For **nulls\_ignored**, **NULL** values are ignored.

The method used to generate table statistics influences how the optimizer chooses indexes for query execution, as described in [Section 8.3.8, “InnoDB and MyISAM Index Statistics Collection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-statistics).

**[innodb\_stats\_on\_metadata](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_on_metadata)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-on-metadata[={OFF|ON}]** |
| **System Variable** | [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This option only applies when optimizer [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) are configured to be non-persistent. Optimizer statistics are not persisted to disk when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is disabled or when individual tables are created or altered with **STATS\_PERSISTENT=0**. For more information, see [Section 15.8.10.2, “Configuring Non-Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation).

When [**innodb\_stats\_on\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_on_metadata) is enabled, **InnoDB** updates non-persistent [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) when metadata statements such as [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) or when accessing the [**INFORMATION\_SCHEMA.TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) or [**INFORMATION\_SCHEMA.STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table) tables. (These updates are similar to what happens for [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table).) When disabled, **InnoDB** does not update statistics during these operations. Leaving the setting disabled can improve access speed for schemas that have a large number of tables or indexes. It can also improve the stability of [execution plans](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query_execution_plan) for queries that involve **InnoDB** tables.

To change the setting, issue the statement **SET GLOBAL innodb\_stats\_on\_metadata=*mode***, where ***mode*** is either **ON** or **OFF** (or **1** or **0**). Changing the setting requires privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)) and immediately affects the operation of all connections.

**[innodb\_stats\_persistent](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_persistent)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-persistent[={OFF|ON}]** |
| **System Variable** | [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies whether **InnoDB** index statistics are persisted to disk. Otherwise, statistics may be recalculated frequently which can lead to variations in [query execution plans](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query_execution_plan). This setting is stored with each table when the table is created. You can set [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) at the global level before creating a table, or use the **STATS\_PERSISTENT** clause of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements to override the system-wide setting and configure persistent statistics for individual tables.

For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**[innodb\_stats\_persistent\_sample\_pages](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_persistent_sample_pages)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-persistent-sample-pages=#** |
| **System Variable** | [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **20** |

The number of index [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) to sample when estimating [cardinality](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cardinality) and other [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) for an indexed column, such as those calculated by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table). Increasing the value improves the accuracy of index statistics, which can improve the [query execution plan](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query_execution_plan), at the expense of increased I/O during the execution of [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) for an **InnoDB** table. For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**Note**

Setting a high value for [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) could result in lengthy [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) execution time. To estimate the number of database pages accessed by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), see [Section 15.8.10.3, “Estimating ANALYZE TABLE Complexity for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-analyze-table-complexity).

[**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) only applies when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is enabled for a table; when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is disabled, [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) applies instead.

**[innodb\_stats\_transient\_sample\_pages](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_stats_transient_sample_pages)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-stats-transient-sample-pages=#** |
| **System Variable** | [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **8** |

The number of index [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) to sample when estimating [cardinality](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cardinality) and other [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) for an indexed column, such as those calculated by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table). The default value is 8. Increasing the value improves the accuracy of index statistics, which can improve the [query execution plan](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query_execution_plan), at the expense of increased I/O when opening an **InnoDB** table or recalculating statistics. For more information, see [Section 15.8.10.2, “Configuring Non-Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation).

**Note**

Setting a high value for [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) could result in lengthy [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) execution time. To estimate the number of database pages accessed by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), see [Section 15.8.10.3, “Estimating ANALYZE TABLE Complexity for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-analyze-table-complexity).

[**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) only applies when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is disabled for a table; when [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is enabled, [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) applies instead. Takes the place of [**innodb\_stats\_sample\_pages**](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_stats_sample_pages). For more information, see [Section 15.8.10.2, “Configuring Non-Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation).

**[innodb\_status\_output](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_status_output)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-status-output[={OFF|ON}]** |
| **System Variable** | [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables or disables periodic output for the standard **InnoDB** Monitor. Also used in combination with [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) to enable or disable periodic output for the **InnoDB** Lock Monitor. For more information, see [Section 15.17.2, “Enabling InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-enabling-monitors).

**[innodb\_status\_output\_locks](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_status_output_locks)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-status-output-locks[={OFF|ON}]** |
| **System Variable** | [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables or disables the **InnoDB** Lock Monitor. When enabled, the **InnoDB** Lock Monitor prints additional information about locks in **SHOW ENGINE INNODB STATUS** output and in periodic output printed to the MySQL error log. Periodic output for the **InnoDB** Lock Monitor is printed as part of the standard **InnoDB** Monitor output. The standard **InnoDB** Monitor must therefore be enabled for the **InnoDB** Lock Monitor to print data to the MySQL error log periodically. For more information, see [Section 15.17.2, “Enabling InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-enabling-monitors).

**[innodb\_strict\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_strict_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-strict-mode[={OFF|ON}]** |
| **System Variable** | [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled, **InnoDB** returns errors rather than warnings for certain conditions.

[Strict mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_strict_mode) helps guard against ignored typos and syntax errors in SQL, or other unintended consequences of various combinations of operational modes and SQL statements. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled, **InnoDB** raises error conditions in certain cases, rather than issuing a warning and processing the specified statement (perhaps with unintended behavior). This is analogous to [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode) in MySQL, which controls what SQL syntax MySQL accepts, and determines whether it silently ignores errors, or validates input syntax and data values.

The [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) setting affects the handling of syntax errors for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), and [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statements. [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) also enables a record size check, so that an **INSERT** or **UPDATE** never fails due to the record being too large for the selected page size.

Oracle recommends enabling [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) when using **ROW\_FORMAT** and **KEY\_BLOCK\_SIZE** clauses in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statements. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, **InnoDB** ignores conflicting clauses and creates the table or index with only a warning in the message log. The resulting table might have different characteristics than intended, such as lack of compression support when attempting to create a compressed table. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled, such problems generate an immediate error and the table or index is not created.

You can enable or disable [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) on the command line when starting **mysqld**, or in a MySQL [configuration file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_configuration_file). You can also enable or disable [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) at runtime with the statement **SET [GLOBAL|SESSION] innodb\_strict\_mode=*mode***, where ***mode*** is either **ON** or **OFF**. Changing the **GLOBAL** setting requires privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)) and affects the operation of all clients that subsequently connect. Any client can change the **SESSION** setting for [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode), and the setting affects only that client.

[**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is not applicable to [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace). Tablespace management rules for general tablespaces are strictly enforced independently of [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode). For more information, see [Section 13.1.21, “CREATE TABLESPACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace).

**[innodb\_sync\_array\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_sync_array_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-sync-array-size=#** |
| **System Variable** | [**innodb\_sync\_array\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_array_size) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1024** |

Defines the size of the mutex/lock wait array. Increasing the value splits the internal data structure used to coordinate threads, for higher concurrency in workloads with large numbers of waiting threads. This setting must be configured when the MySQL instance is starting up, and cannot be changed afterward. Increasing the value is recommended for workloads that frequently produce a large number of waiting threads, typically greater than 768.

**[innodb\_sync\_spin\_loops](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_sync_spin_loops)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-sync-spin-loops=#** |
| **System Variable** | [**innodb\_sync\_spin\_loops**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_spin_loops) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **30** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The number of times a thread waits for an **InnoDB** mutex to be freed before the thread is suspended.

**[innodb\_sync\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_sync_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-sync-debug[={OFF|ON}]** |
| **System Variable** | [**innodb\_sync\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_sync_debug) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables sync debug checking for the **InnoDB** storage engine. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_table\_locks](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_table_locks)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-table-locks[={OFF|ON}]** |
| **System Variable** | [**innodb\_table\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_table_locks) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

If [**autocommit = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), **InnoDB** honors [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables); MySQL does not return from **LOCK TABLES ... WRITE** until all other threads have released all their locks to the table. The default value of [**innodb\_table\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_table_locks) is 1, which means that [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) causes InnoDB to lock a table internally if [**autocommit = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit).

[**innodb\_table\_locks = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_table_locks) has no effect for tables locked explicitly with [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). It does have an effect for tables locked for read or write by [**LOCK TABLES ... WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) implicitly (for example, through triggers) or by [**LOCK TABLES ... READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

For related information, see [Section 15.7, “InnoDB Locking and Transaction Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-transaction-model).

**[innodb\_temp\_data\_file\_path](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_temp_data_file_path)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-temp-data-file-path=file\_name** |
| **System Variable** | [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **ibtmp1:12M:autoextend** |

Defines the relative path, name, size, and attributes of global temporary tablespace data files. The global temporary tablespace stores rollback segments for changes made to user-created temporary tables.

If no value is specified for [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path), the default behavior is to create a single auto-extending data file named ibtmp1 in the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) directory. The initial file size is slightly larger than 12MB.

The syntax for a global temporary tablespace data file specification includes the file name, file size, and **autoextend** and **max** attributes:

***file\_name***:***file\_size***[:autoextend[:max:***max\_file\_size***]]

The global temporary tablespace data file cannot have the same name as another **InnoDB** data file. Any inability or error creating the global temporary tablespace data file is treated as fatal and server startup is refused.

File sizes are specified in KB, MB, or GB by appending **K**, **M** or **G** to the size value. The sum of file sizes must be slightly larger than 12MB.

The size limit of individual files is determined by the operating system. File size can be more than 4GB on operating systems that support large files. Use of raw disk partitions for global temporary tablespace data files is not supported.

The **autoextend** and **max** attributes can be used only for the data file specified last in the [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) setting. For example:

[mysqld]

innodb\_temp\_data\_file\_path=ibtmp1:50M;ibtmp2:12M:autoextend:max:500MB

The **autoextend** option causes the data file to automatically increase in size when it runs out of free space. The **autoextend** increment is 64MB by default. To modify the increment, change the [**innodb\_autoextend\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_autoextend_increment) variable setting.

The directory path for global temporary tablespace data files is formed by concatenating the paths defined by [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) and [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path).

Before running **InnoDB** in read-only mode, set [**innodb\_temp\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_data_file_path) to a location outside of the data directory. The path must be relative to the data directory. For example:

--innodb-temp-data-file-path=../../../tmp/ibtmp1:12M:autoextend

For more information, see [Global Temporary Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-global-temporary-tablespace).

**[innodb\_temp\_tablespaces\_dir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_temp_tablespaces_dir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-temp-tablespaces-dir=dir\_name** |
| **Introduced** | 8.0.13 |
| **System Variable** | [**innodb\_temp\_tablespaces\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_temp_tablespaces_dir) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |
| **Default Value** | **#innodb\_temp** |

Defines the location where **InnoDB** creates a pool of session temporary tablespaces at startup. The default location is the #innodb\_temp directory in the data directory. A fully qualified path or path relative to the data directory is permitted.

As of MySQL 8.0.16, session temporary tablespaces always store user-created temporary tables and internal temporary tables created by the optimizer using **InnoDB**. (Previously, the on-disk storage engine for internal temporary tables was determined by the [**internal\_tmp\_disk\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_internal_tmp_disk_storage_engine) system variable, which is no longer supported. See [Storage Engine for On-Disk Internal Temporary Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-temporary-tables-engines-disk).)

For more information, see [Session Temporary Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-session-temporary-tablespaces).

**[innodb\_thread\_concurrency](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_thread_concurrency)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-thread-concurrency=#** |
| **System Variable** | [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000** |

Defines the maximum number of threads permitted inside of **InnoDB**. A value of 0 (the default) is interpreted as infinite concurrency (no limit). This variable is intended for performance tuning on high concurrency systems.

**InnoDB** tries to keep the number of threads inside **InnoDB** less than or equal to the [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) limit. Once the limit is reached, additional threads are placed into a “First In, First Out” (FIFO) queue for waiting threads. Threads waiting for locks are not counted in the number of concurrently executing threads.

The correct setting depends on workload and computing environment. Consider setting this variable if your MySQL instance shares CPU resources with other applications or if your workload or number of concurrent users is growing. Test a range of values to determine the setting that provides the best performance. [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) is a dynamic variable, which permits experimenting with different settings on a live test system. If a particular setting performs poorly, you can quickly set [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) back to 0.

Use the following guidelines to help find and maintain an appropriate setting:

If the number of concurrent user threads for a workload is consistently small and does not affect performance, set [**innodb\_thread\_concurrency=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) (no limit).

If your workload is consistently heavy or occasionally spikes, set an [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) value and adjust it until you find the number of threads that provides the best performance. For example, suppose that your system typically has 40 to 50 users, but periodically the number increases to 60, 70, or more. Through testing, you find that performance remains largely stable with a limit of 80 concurrent users. In this case, set [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) to 80.

If you do not want **InnoDB** to use more than a certain number of virtual CPUs for user threads (20 virtual CPUs, for example), set [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) to this number (or possibly lower, depending on performance testing). If your goal is to isolate MySQL from other applications, consider binding the **mysqld** process exclusively to the virtual CPUs. Be aware, however, that exclusive binding can result in non-optimal hardware usage if the **mysqld** process is not consistently busy. In this case, you can bind the **mysqld** process to the virtual CPUs but allow other applications to use some or all of the virtual CPUs.

**Note**

From an operating system perspective, using a resource management solution to manage how CPU time is shared among applications may be preferable to binding the **mysqld** process. For example, you could assign 90% of virtual CPU time to a given application while other critical processes are not running, and scale that value back to 40% when other critical processes are running.

In some cases, the optimal [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) setting can be smaller than the number of virtual CPUs.

An [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) value that is too high can cause performance regression due to increased contention on system internals and resources.

Monitor and analyze your system regularly. Changes to workload, number of users, or computing environment may require that you adjust the [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) setting.

A value of 0 disables the **queries inside InnoDB** and **queries in queue**counters in the **ROW OPERATIONS** section of **SHOW ENGINE INNODB STATUS** output.

For related information, see [Section 15.8.4, “Configuring Thread Concurrency for InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-thread_concurrency).

**[innodb\_thread\_sleep\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_thread_sleep_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-thread-sleep-delay=#** |
| **System Variable** | [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000000** |

How long **InnoDB** threads sleep before joining the **InnoDB** queue, in microseconds. The default value is 10000. A value of 0 disables sleep. You can set [**innodb\_adaptive\_max\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_max_sleep_delay) to the highest value you would allow for [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay), and **InnoDB** automatically adjusts [**innodb\_thread\_sleep\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_sleep_delay) up or down depending on current thread-scheduling activity. This dynamic adjustment helps the thread scheduling mechanism to work smoothly during times when the system is lightly loaded or when it is operating near full capacity.

For more information, see [Section 15.8.4, “Configuring Thread Concurrency for InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-thread_concurrency).

**[innodb\_tmpdir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_tmpdir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-tmpdir=dir\_name** |
| **System Variable** | [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |
| **Default Value** | **NULL** |

Used to define an alternate directory for temporary sort files created during online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that rebuild the table.

Online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that rebuild the table also create an intermediate table file in the same directory as the original table. The [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) option is not applicable to intermediate table files.

A valid value is any directory path other than the MySQL data directory path. If the value is NULL (the default), temporary files are created MySQL temporary directory (**$TMPDIR** on Unix, **%TEMP%** on Windows, or the directory specified by the [**--tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) configuration option). If a directory is specified, existence of the directory and permissions are only checked when [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) is configured using a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. If a symlink is provided in a directory string, the symlink is resolved and stored as an absolute path. The path should not exceed 512 bytes. An online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation reports an error if [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) is set to an invalid directory. [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) overrides the MySQL [**tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) setting but only for online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations.

The **FILE** privilege is required to configure [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir).

The [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) option was introduced to help avoid overflowing a temporary file directory located on a **tmpfs** file system. Such overflows could occur as a result of large temporary sort files created during online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that rebuild the table.

In replication environments, only consider replicating the [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) setting if all servers have the same operating system environment. Otherwise, replicating the [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) setting could result in a replication failure when running online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that rebuild the table. If server operating environments differ, it is recommended that you configure [**innodb\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_tmpdir) on each server individually.

For more information, see [Section 15.12.3, “Online DDL Space Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-space-requirements). For information about online [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations, see [Section 15.12, “InnoDB and Online DDL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl).

**[innodb\_trx\_purge\_view\_update\_only\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_trx_purge_view_update_only_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-trx-purge-view-update-only-debug[={OFF|ON}]** |
| **System Variable** | [**innodb\_trx\_purge\_view\_update\_only\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_trx_purge_view_update_only_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Pauses purging of delete-marked records while allowing the purge view to be updated. This option artificially creates a situation in which the purge view is updated but purges have not yet been performed. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_trx\_rseg\_n\_slots\_debug](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_trx_rseg_n_slots_debug)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-trx-rseg-n-slots-debug=#** |
| **System Variable** | [**innodb\_trx\_rseg\_n\_slots\_debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_trx_rseg_n_slots_debug) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Maximum Value** | **1024** |

Sets a debug flag that limits **TRX\_RSEG\_N\_SLOTS** to a given value for the **trx\_rsegf\_undo\_find\_free** function that looks for free slots for undo log segments. This option is only available if debugging support is compiled in using the [WITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug) **CMake** option.

**[innodb\_undo\_directory](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_undo_directory)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-undo-directory=dir\_name** |
| **System Variable** | [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |

The path where **InnoDB** creates undo tablespaces. Typically used to place undo tablespaces on a different storage device.

There is no default value (it is NULL). If the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is undefined, undo tablespaces are created in the data directory.

The default undo tablespaces (innodb\_undo\_001 and innodb\_undo\_002) created when the MySQL instance is initialized always reside in the directory defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable.

Undo tablespaces created using [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax are created in the directory defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable if a different path is not specified.

For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

**[innodb\_undo\_log\_encrypt](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_undo_log_encrypt)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-undo-log-encrypt[={OFF|ON}]** |
| **System Variable** | [**innodb\_undo\_log\_encrypt**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_encrypt) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Controls encryption of undo log data for tables encrypted using the **InnoDB** [data-at-rest encryption feature](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption). Only applies to undo logs that reside in separate [undo tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace). See [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces). Encryption is not supported for undo log data that resides in the system tablespace. For more information, see [Undo Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-undo-log).

**[innodb\_undo\_log\_truncate](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_undo_log_truncate)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-undo-log-truncate[={OFF|ON}]** |
| **System Variable** | [**innodb\_undo\_log\_truncate**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_log_truncate) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

When enabled, undo tablespaces that exceed the threshold value defined by [**innodb\_max\_undo\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_max_undo_log_size) are marked for truncation. Only undo tablespaces can be truncated. Truncating undo logs that reside in the system tablespace is not supported. For truncation to occur, there must be at least two undo tablespaces.

The [**innodb\_purge\_rseg\_truncate\_frequency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_purge_rseg_truncate_frequency) variable can be used to expedite truncation of undo tablespaces.

For more information, see [Truncating Undo Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#truncate-undo-tablespace).

**[innodb\_undo\_tablespaces](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_undo_tablespaces)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-undo-tablespaces=#** |
| **Deprecated** | Yes |
| **System Variable** | [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2** |
| **Minimum Value** | **2** |
| **Maximum Value** | **127** |

Defines the number of [undo tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace) used by **InnoDB**. The default and minimum value is 2.

**Note**

The [**innodb\_undo\_tablespaces**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_tablespaces) variable is deprecated and is no longer configurable as of MySQL 8.0.14. Expect it to be removed in a future release.

For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

**[innodb\_use\_fdatasync](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_use_fdatasync)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-use-fdatasync[={OFF|ON}]** |
| **Introduced** | 8.0.25 |
| **System Variable** | [**innodb\_use\_fdatasync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_fdatasync) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

When enabled, **InnoDB** uses **fdatasync()** instead of **fsync()** when flushing data to the operating system. Unlike **fsync()**, which is used by default, **fdatasync()** only flushes the metadata of accessed files as necessary, providing a performance benefit in certain scenarios.

**[innodb\_use\_native\_aio](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_use_native_aio)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-use-native-aio[={OFF|ON}]** |
| **System Variable** | [**innodb\_use\_native\_aio**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_native_aio) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Specifies whether to use the Linux asynchronous I/O subsystem. This variable applies to Linux systems only, and cannot be changed while the server is running. Normally, you do not need to configure this option, because it is enabled by default.

The [asynchronous I/O](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_asynchronous_io) capability that **InnoDB** has on Windows systems is available on Linux systems. (Other Unix-like systems continue to use synchronous I/O calls.) This feature improves the scalability of heavily I/O-bound systems, which typically show many pending reads/writes in **SHOW ENGINE INNODB STATUS\G** output.

Running with a large number of **InnoDB** I/O threads, and especially running multiple such instances on the same server machine, can exceed capacity limits on Linux systems. In this case, you may receive the following error:

EAGAIN: The specified maxevents exceeds the user's limit of available events.

You can typically address this error by writing a higher limit to **/proc/sys/fs/aio-max-nr**.

However, if a problem with the asynchronous I/O subsystem in the OS prevents **InnoDB** from starting, you can start the server with [**innodb\_use\_native\_aio=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_use_native_aio). This option may also be disabled automatically during startup if **InnoDB** detects a potential problem such as a combination of **tmpdir** location, **tmpfs** file system, and Linux kernel that does not support AIO on **tmpfs**.

For more information, see [Section 15.8.6, “Using Asynchronous I/O on Linux”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-linux-native-aio).

**innodb\_validate\_tablespace\_paths**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-validate-tablespace-paths[={OFF|ON}]** |
| **Introduced** | 8.0.21 |
| **System Variable** | [**innodb\_validate\_tablespace\_paths**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_validate_tablespace_paths) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Controls tablespace file path validation. At startup, **InnoDB** validates the paths of known tablespace files against tablespace file paths stored in the data dictionary in case tablespace files have been moved to a different location. The [**innodb\_validate\_tablespace\_paths**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_validate_tablespace_paths) variable permits disabling tablespace path validation. This feature is intended for environments where tablespaces files are not moved. Disabling path validation improves startup time on systems with a large number of tablespace files.

**Warning**

Starting the server with tablespace path validation disabled after moving tablespace files can lead to undefined behavior.

For more information, see [Section 15.6.3.7, “Disabling Tablespace Path Validation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disabling-tablespace-path-validation).

**[innodb\_version](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_version)**

The **InnoDB** version number. In MySQL 8.0, separate version numbering for **InnoDB** does not apply and this value is the same the [**version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_version) number of the server.

**[innodb\_write\_io\_threads](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\innodb-storage-engine.html" \l "sysvar_innodb_write_io_threads)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--innodb-write-io-threads=#** |
| **System Variable** | [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **4** |
| **Minimum Value** | **1** |
| **Maximum Value** | **64** |

The number of I/O threads for write operations in **InnoDB**. The default value is 4. Its counterpart for read threads is [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads). For more information, see [Section 15.8.5, “Configuring the Number of Background InnoDB I/O Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-performance-multiple_io_threads). For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

**Note**

On Linux systems, running multiple MySQL servers (typically more than 12) with default settings for [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads), [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads), and the Linux **aio-max-nr** setting can exceed system limits. Ideally, increase the **aio-max-nr** setting; as a workaround, you might reduce the settings for one or both of the MySQL variables.

Also take into consideration the value of [**sync\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog), which controls synchronization of the binary log to disk.

For general I/O tuning advice, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

## 15.15 InnoDB INFORMATION\_SCHEMA Tables

[15.15.1 InnoDB INFORMATION\_SCHEMA Tables about Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-compression-tables)

[15.15.2 InnoDB INFORMATION\_SCHEMA Transaction and Locking Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-transactions)

[15.15.3 InnoDB INFORMATION\_SCHEMA Schema Object Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-system-tables)

[15.15.4 InnoDB INFORMATION\_SCHEMA FULLTEXT Index Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-fulltext_index-tables)

[15.15.5 InnoDB INFORMATION\_SCHEMA Buffer Pool Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-buffer-pool-tables)

[15.15.6 InnoDB INFORMATION\_SCHEMA Metrics Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-metrics-table)

[15.15.7 InnoDB INFORMATION\_SCHEMA Temporary Table Info Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-temp-table-info)

[15.15.8 Retrieving InnoDB Tablespace Metadata from INFORMATION\_SCHEMA.FILES](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-files-table)

This section provides information and usage examples for **InnoDB** [**INFORMATION\_SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html) tables.

**InnoDB** **INFORMATION\_SCHEMA** tables provide metadata, status information, and statistics about various aspects of the **InnoDB** storage engine. You can view a list of **InnoDB** **INFORMATION\_SCHEMA** tables by issuing a [**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables) statement on the **INFORMATION\_SCHEMA** database:

mysql> **SHOW TABLES FROM INFORMATION\_SCHEMA LIKE 'INNODB%';**

For table definitions, see [Section 26.4, “INFORMATION\_SCHEMA InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#innodb-information-schema-tables). For general information regarding the **MySQL** **INFORMATION\_SCHEMA** database, see [Chapter 26, *INFORMATION\_SCHEMA Tables*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html).

### 15.15.1 InnoDB INFORMATION\_SCHEMA Tables about Compression

[15.15.1.1 INNODB\_CMP and INNODB\_CMP\_RESET](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-innodb_cmp)

[15.15.1.2 INNODB\_CMPMEM and INNODB\_CMPMEM\_RESET](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-innodb_cmpmem)

[15.15.1.3 Using the Compression Information Schema Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-examples-compression-sect)

There are two pairs of **InnoDB** **INFORMATION\_SCHEMA** tables about compression that can provide insight into how well compression is working overall:

[**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) and [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) provide information about the number of compression operations and the amount of time spent performing compression.

[**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) and [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) provide information about the way memory is allocated for compression.

#### 15.15.1.1 INNODB\_CMP and INNODB\_CMP\_RESET

The [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) and [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) tables provide status information about operations related to compressed tables, which are described in [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression). The **PAGE\_SIZE** column reports the compressed [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size).

These two tables have identical contents, but reading from [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) resets the statistics on compression and uncompression operations. For example, if you archive the output of [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) every 60 minutes, you see the statistics for each hourly period. If you monitor the output of [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) (making sure never to read [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table)), you see the cumulative statistics since InnoDB was started.

For the table definition, see [Section 26.4.6, “The INFORMATION\_SCHEMA INNODB\_CMP and INNODB\_CMP\_RESET Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table).

#### 15.15.1.2 INNODB\_CMPMEM and INNODB\_CMPMEM\_RESET

The [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) and [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) tables provide status information about compressed pages that reside in the buffer pool. Please consult [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression) for further information on compressed tables and the use of the buffer pool. The [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) and [**INNODB\_CMP\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) tables should provide more useful statistics on compression.

##### Internal Details

**InnoDB** uses a [buddy allocator](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buddy_allocator) system to manage memory allocated to [pages of various sizes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size), from 1KB to 16KB. Each row of the two tables described here corresponds to a single page size.

The [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) and [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) tables have identical contents, but reading from [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) resets the statistics on relocation operations. For example, if every 60 minutes you archived the output of [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table), it would show the hourly statistics. If you never read [**INNODB\_CMPMEM\_RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) and monitored the output of [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) instead, it would show the cumulative statistics since **InnoDB** was started.

For the table definition, see [Section 26.4.7, “The INFORMATION\_SCHEMA INNODB\_CMPMEM and INNODB\_CMPMEM\_RESET Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table).

#### 15.15.1.3 Using the Compression Information Schema Tables

**Example 15.1 Using the Compression Information Schema Tables**

The following is sample output from a database that contains compressed tables (see [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression), [**INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table), [**INNODB\_CMP\_PER\_INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-per-index-table), and [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table)).

The following table shows the contents of [**INFORMATION\_SCHEMA.INNODB\_CMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmp-table) under a light [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload). The only compressed page size that the buffer pool contains is 8K. Compressing or uncompressing pages has consumed less than a second since the time the statistics were reset, because the columns **COMPRESS\_TIME** and **UNCOMPRESS\_TIME** are zero.

| **page size** | **compress ops** | **compress ops ok** | **compress time** | **uncompress ops** | **uncompress time** |
| --- | --- | --- | --- | --- | --- |
| **1024** | 0 | 0 | 0 | 0 | 0 |
| **2048** | 0 | 0 | 0 | 0 | 0 |
| **4096** | 0 | 0 | 0 | 0 | 0 |
| **8192** | 1048 | 921 | 0 | 61 | 0 |
| **16384** | 0 | 0 | 0 | 0 | 0 |

According to [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table), there are 6169 compressed 8KB pages in the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). The only other allocated block size is 64 bytes. The smallest **PAGE\_SIZE** in [**INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) is used for block descriptors of those compressed pages for which no uncompressed page exists in the buffer pool. We see that there are 5910 such pages. Indirectly, we see that 259 (6169-5910) compressed pages also exist in the buffer pool in uncompressed form.

The following table shows the contents of [**INFORMATION\_SCHEMA.INNODB\_CMPMEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-cmpmem-table) under a light [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload). Some memory is unusable due to fragmentation of the memory allocator for compressed pages: **SUM(PAGE\_SIZE\*PAGES\_FREE)=6784**. This is because small memory allocation requests are fulfilled by splitting bigger blocks, starting from the 16K blocks that are allocated from the main buffer pool, using the buddy allocation system. The fragmentation is this low because some allocated blocks have been relocated (copied) to form bigger adjacent free blocks. This copying of **SUM(PAGE\_SIZE\*RELOCATION\_OPS)** bytes has consumed less than a second **(SUM(RELOCATION\_TIME)=0)**.

| **page size** | **pages used** | **pages free** | **relocation ops** | **relocation time** |
| --- | --- | --- | --- | --- |
| **64** | 5910 | 0 | 2436 | 0 |
| **128** | 0 | 1 | 0 | 0 |
| **256** | 0 | 0 | 0 | 0 |
| **512** | 0 | 1 | 0 | 0 |
| **1024** | 0 | 0 | 0 | 0 |
| **2048** | 0 | 1 | 0 | 0 |
| **4096** | 0 | 1 | 0 | 0 |
| **8192** | 6169 | 0 | 5 | 0 |
| **16384** | 0 | 0 | 0 | 0 |

### 15.15.2 InnoDB INFORMATION\_SCHEMA Transaction and Locking Information

[15.15.2.1 Using InnoDB Transaction and Locking Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-examples)

[15.15.2.2 InnoDB Lock and Lock-Wait Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-understanding-innodb-locking)

[15.15.2.3 Persistence and Consistency of InnoDB Transaction and Locking Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-internal-data)

**Note**

This section describes locking information as exposed by the Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables, which supersede the **INFORMATION\_SCHEMA** [**INNODB\_LOCKS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-locks-table) and [**INNODB\_LOCK\_WAITS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-lock-waits-table) tables in MySQL 8.0. For similar discussion written in terms of the older **INFORMATION\_SCHEMA** tables, see [InnoDB INFORMATION\_SCHEMA Transaction and Locking Information](https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-transactions.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

One **INFORMATION\_SCHEMA** table and two Performance Schema tables enable you to monitor **InnoDB** transactions and diagnose potential locking problems:

[**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table): This **INFORMATION\_SCHEMA** table provides information about every transaction currently executing inside **InnoDB**, including the transaction state (for example, whether it is running or waiting for a lock), when the transaction started, and the particular SQL statement the transaction is executing.

**[data\_locks](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\performance-schema.html" \l "performance-schema-data-locks-table" \o "27.12.13.1 The data_locks Table)**: This Performance Schema table contains a row for each hold lock and each lock request that is blocked waiting for a held lock to be released:

There is one row for each held lock, whatever the state of the transaction that holds the lock (**INNODB\_TRX.TRX\_STATE** is **RUNNING**, **LOCK WAIT**, **ROLLING BACK** or **COMMITTING**).

Each transaction in InnoDB that is waiting for another transaction to release a lock (**INNODB\_TRX.TRX\_STATE** is **LOCK WAIT**) is blocked by exactly one blocking lock request. That blocking lock request is for a row or table lock held by another transaction in an incompatible mode. A lock request always has a mode that is incompatible with the mode of the held lock that blocks the request (read vs. write, shared vs. exclusive).

The blocked transaction cannot proceed until the other transaction commits or rolls back, thereby releasing the requested lock. For every blocked transaction, [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) contains one row that describes each lock the transaction has requested, and for which it is waiting.

**[data\_lock\_waits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\performance-schema.html" \l "performance-schema-data-lock-waits-table" \o "27.12.13.2 The data_lock_waits Table)**: This Performance Schema table indicates which transactions are waiting for a given lock, or for which lock a given transaction is waiting. This table contains one or more rows for each blocked transaction, indicating the lock it has requested and any locks that are blocking that request. The **REQUESTING\_ENGINE\_LOCK\_ID** value refers to the lock requested by a transaction, and the **BLOCKING\_ENGINE\_LOCK\_ID** value refers to the lock (held by another transaction) that prevents the first transaction from proceeding. For any given blocked transaction, all rows in [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) have the same value for **REQUESTING\_ENGINE\_LOCK\_ID** and different values for **BLOCKING\_ENGINE\_LOCK\_ID**.

For more information about the preceding tables, see [Section 26.4.30, “The INFORMATION\_SCHEMA INNODB\_TRX Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table), [Section 27.12.13.1, “The data\_locks Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table), and [Section 27.12.13.2, “The data\_lock\_waits Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table).

#### 15.15.2.1 Using InnoDB Transaction and Locking Information

**Note**

This section describes locking information as exposed by the Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables, which supersede the **INFORMATION\_SCHEMA** [**INNODB\_LOCKS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-locks-table) and [**INNODB\_LOCK\_WAITS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-lock-waits-table) tables in MySQL 8.0. For similar discussion written in terms of the older **INFORMATION\_SCHEMA** tables, see [Using InnoDB Transaction and Locking Information](https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-examples.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

##### Identifying Blocking Transactions

It is sometimes helpful to identify which transaction blocks another. The tables that contain information about **InnoDB** transactions and data locks enable you to determine which transaction is waiting for another, and which resource is being requested. (For descriptions of these tables, see [Section 15.15.2, “InnoDB INFORMATION\_SCHEMA Transaction and Locking Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-transactions).)

Suppose that three sessions are running concurrently. Each session corresponds to a MySQL thread, and executes one transaction after another. Consider the state of the system when these sessions have issued the following statements, but none has yet committed its transaction:

Session A:

BEGIN;

SELECT a FROM t FOR UPDATE;

SELECT SLEEP(100);

Session B:

SELECT b FROM t FOR UPDATE;

Session C:

SELECT c FROM t FOR UPDATE;

In this scenario, use the following query to see which transactions are waiting and which transactions are blocking them:

SELECT

r.trx\_id waiting\_trx\_id,

r.trx\_mysql\_thread\_id waiting\_thread,

r.trx\_query waiting\_query,

b.trx\_id blocking\_trx\_id,

b.trx\_mysql\_thread\_id blocking\_thread,

b.trx\_query blocking\_query

FROM performance\_schema.data\_lock\_waits w

INNER JOIN information\_schema.innodb\_trx b

ON b.trx\_id = w.blocking\_engine\_transaction\_id

INNER JOIN information\_schema.innodb\_trx r

ON r.trx\_id = w.requesting\_engine\_transaction\_id;

Or, more simply, use the **sys** schema [**innodb\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-innodb-lock-waits) view:

SELECT

waiting\_trx\_id,

waiting\_pid,

waiting\_query,

blocking\_trx\_id,

blocking\_pid,

blocking\_query

FROM sys.innodb\_lock\_waits;

If a NULL value is reported for the blocking query, see [Identifying a Blocking Query After the Issuing Session Becomes Idle](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-examples-null-blocking-query).

| **waiting trx id** | **waiting thread** | **waiting query** | **blocking trx id** | **blocking thread** | **blocking query** |
| --- | --- | --- | --- | --- | --- |
| **A4** | **6** | **SELECT b FROM t FOR UPDATE** | **A3** | **5** | **SELECT SLEEP(100)** |
| **A5** | **7** | **SELECT c FROM t FOR UPDATE** | **A3** | **5** | **SELECT SLEEP(100)** |
| **A5** | **7** | **SELECT c FROM t FOR UPDATE** | **A4** | **6** | **SELECT b FROM t FOR UPDATE** |

In the preceding table, you can identify sessions by the “waiting query” or “blocking query” columns. As you can see:

Session B (trx id **A4**, thread **6**) and Session C (trx id **A5**, thread **7**) are both waiting for Session A (trx id **A3**, thread **5**).

Session C is waiting for Session B as well as Session A.

You can see the underlying data in the **INFORMATION\_SCHEMA** [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table and Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables.

The following table shows some sample contents of the [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table.

| **trx id** | **trx state** | **trx started** | **trx requested lock id** | **trx wait started** | **trx weight** | **trx mysql thread id** | **trx query** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A3** | **RUN­NING** | **2008-01-15 16:44:54** | **NULL** | **NULL** | **2** | **5** | **SELECT SLEEP(100)** |
| **A4** | **LOCK WAIT** | **2008-01-15 16:45:09** | **A4:1:3:2** | **2008-01-15 16:45:09** | **2** | **6** | **SELECT b FROM t FOR UPDATE** |
| **A5** | **LOCK WAIT** | **2008-01-15 16:45:14** | **A5:1:3:2** | **2008-01-15 16:45:14** | **2** | **7** | **SELECT c FROM t FOR UPDATE** |

The following table shows some sample contents of the [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) table.

| **lock id** | **lock trx id** | **lock mode** | **lock type** | **lock schema** | **lock table** | **lock index** | **lock data** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A3:1:3:2** | **A3** | **X** | **RECORD** | **test** | **t** | **PRIMARY** | **0x0200** |
| **A4:1:3:2** | **A4** | **X** | **RECORD** | **test** | **t** | **PRIMARY** | **0x0200** |
| **A5:1:3:2** | **A5** | **X** | **RECORD** | **test** | **t** | **PRIMARY** | **0x0200** |

The following table shows some sample contents of the [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) table.

| **requesting trx id** | **requested lock id** | **blocking trx id** | **blocking lock id** |
| --- | --- | --- | --- |
| **A4** | **A4:1:3:2** | **A3** | **A3:1:3:2** |
| **A5** | **A5:1:3:2** | **A3** | **A3:1:3:2** |
| **A5** | **A5:1:3:2** | **A4** | **A4:1:3:2** |

##### Identifying a Blocking Query After the Issuing Session Becomes Idle

When identifying blocking transactions, a NULL value is reported for the blocking query if the session that issued the query has become idle. In this case, use the following steps to determine the blocking query:

Identify the processlist ID of the blocking transaction. In the [**sys.innodb\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sys-schema.html#sys-innodb-lock-waits) table, the processlist ID of the blocking transaction is the **blocking\_pid** value.

Using the **blocking\_pid**, query the MySQL Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table to determine the **THREAD\_ID** of the blocking transaction. For example, if the **blocking\_pid** is 6, issue this query:

SELECT THREAD\_ID FROM performance\_schema.threads WHERE PROCESSLIST\_ID = 6;

Using the **THREAD\_ID**, query the Performance Schema [**events\_statements\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-statements-current-table) table to determine the last query executed by the thread. For example, if the **THREAD\_ID** is 28, issue this query:

SELECT THREAD\_ID, SQL\_TEXT FROM performance\_schema.events\_statements\_current

WHERE THREAD\_ID = 28\G

If the last query executed by the thread is not enough information to determine why a lock is held, you can query the Performance Schema [**events\_statements\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-statements-history-table) table to view the last 10 statements executed by the thread.

SELECT THREAD\_ID, SQL\_TEXT FROM performance\_schema.events\_statements\_history

WHERE THREAD\_ID = 28 ORDER BY EVENT\_ID;

##### Correlating InnoDB Transactions with MySQL Sessions

Sometimes it is useful to correlate internal **InnoDB** locking information with the session-level information maintained by MySQL. For example, you might like to know, for a given **InnoDB** transaction ID, the corresponding MySQL session ID and name of the session that may be holding a lock, and thus blocking other transactions.

The following output from the **INFORMATION\_SCHEMA** [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table and Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables is taken from a somewhat loaded system. As can be seen, there are several transactions running.

The following [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables show that:

Transaction **77F** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transactions **77E**, **77D**, and **77B** to commit.

Transaction **77E** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transactions **77D** and **77B** to commit.

Transaction **77D** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transaction **77B** to commit.

Transaction **77B** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transaction **77A** to commit.

Transaction **77A** is running, currently executing [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

Transaction **E56** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transaction **E55** to commit.

Transaction **E55** (executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)) is waiting for transaction **19C** to commit.

Transaction **19C** is running, currently executing an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert).

**Note**

There may be inconsistencies between queries shown in the **INFORMATION\_SCHEMA** [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) and [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) tables. For an explanation, see [Section 15.15.2.3, “Persistence and Consistency of InnoDB Transaction and Locking Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-internal-data).

The following table shows the contents of the [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table for a system running a heavy [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload).

| **ID** | **USER** | **HOST** | **DB** | **COMMAND** | **TIME** | **STATE** | **INFO** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **384** | **root** | **localhost** | **test** | **Query** | **10** | **update** | **INSERT INTO t2 VALUES …** |
| **257** | **root** | **localhost** | **test** | **Query** | **3** | **update** | **INSERT INTO t2 VALUES …** |
| **130** | **root** | **localhost** | **test** | **Query** | **0** | **update** | **INSERT INTO t2 VALUES …** |
| **61** | **root** | **localhost** | **test** | **Query** | **1** | **update** | **INSERT INTO t2 VALUES …** |
| **8** | **root** | **localhost** | **test** | **Query** | **1** | **update** | **INSERT INTO t2 VALUES …** |
| **4** | **root** | **localhost** | **test** | **Query** | **0** | **preparing** | **SELECT \* FROM PROCESSLIST** |
| **2** | **root** | **localhost** | **test** | **Sleep** | **566** |  | **NULL** |

The following table shows the contents of the [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table for a system running a heavy [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload).

| **trx id** | **trx state** | **trx started** | **trx requested lock id** | **trx wait started** | **trx weight** | **trx mysql thread id** | **trx query** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **77F** | **LOCK WAIT** | **2008-01-15 13:10:16** | **77F** | **2008-01-15 13:10:16** | **1** | **876** | **INSERT INTO t09 (D, B, C) VALUES …** |
| **77E** | **LOCK WAIT** | **2008-01-15 13:10:16** | **77E** | **2008-01-15 13:10:16** | **1** | **875** | **INSERT INTO t09 (D, B, C) VALUES …** |
| **77D** | **LOCK WAIT** | **2008-01-15 13:10:16** | **77D** | **2008-01-15 13:10:16** | **1** | **874** | **INSERT INTO t09 (D, B, C) VALUES …** |
| **77B** | **LOCK WAIT** | **2008-01-15 13:10:16** | **77B:733:12:1** | **2008-01-15 13:10:16** | **4** | **873** | **INSERT INTO t09 (D, B, C) VALUES …** |
| **77A** | **RUN­NING** | **2008-01-15 13:10:16** | **NULL** | **NULL** | **4** | **872** | **SELECT b, c FROM t09 WHERE …** |
| **E56** | **LOCK WAIT** | **2008-01-15 13:10:06** | **E56:743:6:2** | **2008-01-15 13:10:06** | **5** | **384** | **INSERT INTO t2 VALUES …** |
| **E55** | **LOCK WAIT** | **2008-01-15 13:10:06** | **E55:743:38:2** | **2008-01-15 13:10:13** | **965** | **257** | **INSERT INTO t2 VALUES …** |
| **19C** | **RUN­NING** | **2008-01-15 13:09:10** | **NULL** | **NULL** | **2900** | **130** | **INSERT INTO t2 VALUES …** |
| **E15** | **RUN­NING** | **2008-01-15 13:08:59** | **NULL** | **NULL** | **5395** | **61** | **INSERT INTO t2 VALUES …** |
| **51D** | **RUN­NING** | **2008-01-15 13:08:47** | **NULL** | **NULL** | **9807** | **8** | **INSERT INTO t2 VALUES …** |

The following table shows the contents of the [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) table for a system running a heavy [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload).

| **requesting trx id** | **requested lock id** | **blocking trx id** | **blocking lock id** |
| --- | --- | --- | --- |
| **77F** | **77F:806** | **77E** | **77E:806** |
| **77F** | **77F:806** | **77D** | **77D:806** |
| **77F** | **77F:806** | **77B** | **77B:806** |
| **77E** | **77E:806** | **77D** | **77D:806** |
| **77E** | **77E:806** | **77B** | **77B:806** |
| **77D** | **77D:806** | **77B** | **77B:806** |
| **77B** | **77B:733:12:1** | **77A** | **77A:733:12:1** |
| **E56** | **E56:743:6:2** | **E55** | **E55:743:6:2** |
| **E55** | **E55:743:38:2** | **19C** | **19C:743:38:2** |

The following table shows the contents of the [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) table for a system running a heavy [workload](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_workload).

| **lock id** | **lock trx id** | **lock mode** | **lock type** | **lock schema** | **lock table** | **lock index** | **lock data** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **77F:806** | **77F** | **AUTO\_INC** | **TABLE** | **test** | **t09** | **NULL** | **NULL** |
| **77E:806** | **77E** | **AUTO\_INC** | **TABLE** | **test** | **t09** | **NULL** | **NULL** |
| **77D:806** | **77D** | **AUTO\_INC** | **TABLE** | **test** | **t09** | **NULL** | **NULL** |
| **77B:806** | **77B** | **AUTO\_INC** | **TABLE** | **test** | **t09** | **NULL** | **NULL** |
| **77B:733:12:1** | **77B** | **X** | **RECORD** | **test** | **t09** | **PRIMARY** | **supremum pseudo-record** |
| **77A:733:12:1** | **77A** | **X** | **RECORD** | **test** | **t09** | **PRIMARY** | **supremum pseudo-record** |
| **E56:743:6:2** | **E56** | **S** | **RECORD** | **test** | **t2** | **PRIMARY** | **0, 0** |
| **E55:743:6:2** | **E55** | **X** | **RECORD** | **test** | **t2** | **PRIMARY** | **0, 0** |
| **E55:743:38:2** | **E55** | **S** | **RECORD** | **test** | **t2** | **PRIMARY** | **1922, 1922** |
| **19C:743:38:2** | **19C** | **X** | **RECORD** | **test** | **t2** | **PRIMARY** | **1922, 1922** |

#### 15.15.2.2 InnoDB Lock and Lock-Wait Information

**Note**

This section describes locking information as exposed by the Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables, which supersede the **INFORMATION\_SCHEMA** [**INNODB\_LOCKS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-locks-table) and [**INNODB\_LOCK\_WAITS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-lock-waits-table) tables in MySQL 8.0. For similar discussion written in terms of the older **INFORMATION\_SCHEMA** tables, see [InnoDB Lock and Lock-Wait Information](https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-understanding-innodb-locking.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

When a transaction updates a row in a table, or locks it with **SELECT FOR UPDATE**, **InnoDB** establishes a list or queue of locks on that row. Similarly, **InnoDB** maintains a list of locks on a table for table-level locks. If a second transaction wants to update a row or lock a table already locked by a prior transaction in an incompatible mode, **InnoDB** adds a lock request for the row to the corresponding queue. For a lock to be acquired by a transaction, all incompatible lock requests previously entered into the lock queue for that row or table must be removed (which occurs when the transactions holding or requesting those locks either commit or roll back).

A transaction may have any number of lock requests for different rows or tables. At any given time, a transaction may request a lock that is held by another transaction, in which case it is blocked by that other transaction. The requesting transaction must wait for the transaction that holds the blocking lock to commit or roll back. If a transaction is not waiting for a lock, it is in a **RUNNING** state. If a transaction is waiting for a lock, it is in a **LOCK WAIT** state. (The **INFORMATION\_SCHEMA** [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table indicates transaction state values.)

The Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) table holds one or more rows for each **LOCK WAIT** transaction, indicating any lock requests that prevent its progress. This table also contains one row describing each lock in a queue of locks pending for a given row or table. The Performance Schema [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) table shows which locks already held by a transaction are blocking locks requested by other transactions.

#### 15.15.2.3 Persistence and Consistency of InnoDB Transaction and Locking Information

**Note**

This section describes locking information as exposed by the Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables, which supersede the **INFORMATION\_SCHEMA** [**INNODB\_LOCKS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-locks-table) and [**INNODB\_LOCK\_WAITS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-lock-waits-table) tables in MySQL 8.0. For similar discussion written in terms of the older **INFORMATION\_SCHEMA** tables, see [Persistence and Consistency of InnoDB Transaction and Locking Information](https://dev.mysql.com/doc/refman/5.7/en/innodb-information-schema-internal-data.html), in [MySQL 5.7 Reference Manual](https://dev.mysql.com/doc/refman/5.7/en/).

The data exposed by the transaction and locking tables (**INFORMATION\_SCHEMA** [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table, Performance Schema [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables) represents a glimpse into fast-changing data. This is not like user tables, where the data changes only when application-initiated updates occur. The underlying data is internal system-managed data, and can change very quickly:

Data might not be consistent between the [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table), [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table), and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables.

The [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) and [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) tables expose live data from the **InnoDB** storage engine, to provide lock inormation about the transactions in the [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) table. Data retrieved from the lock tables exists when the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is executed, but might be gone or changed by the time the query result is consumed by the client.

Joining [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) with [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) can show rows in [**data\_lock\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-lock-waits-table) that identify a parent row in [**data\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-data-locks-table) that no longer exists or does not exist yet.

Data in the transaction and locking tables might not be consistent with data in the **INFORMATION\_SCHEMA** [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table or Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table.

For example, you should be careful when comparing data in the **InnoDB** transaction and locking tables with data in the [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table. Even if you issue a single **SELECT** (joining [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) and [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table), for example), the content of those tables is generally not consistent. It is possible for [**INNODB\_TRX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-trx-table) to reference rows that are not present in [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) or for the currently executing SQL query of a transaction shown in **INNODB\_TRX.TRX\_QUERY** to differ from the one in **PROCESSLIST.INFO**.

### 15.15.3 InnoDB INFORMATION\_SCHEMA Schema Object Tables

You can extract metadata about schema objects managed by **InnoDB** using **InnoDB** **INFORMATION\_SCHEMA** tables. This information comes from the data dictionary. Traditionally, you would get this type of information using the techniques from [Section 15.17, “InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-monitors), setting up **InnoDB** monitors and parsing the output from the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) statement. The **InnoDB** **INFORMATION\_SCHEMA** table interface allows you to query this data using SQL.

**InnoDB** **INFORMATION\_SCHEMA** schema object tables include the tables listed below.

INNODB\_DATAFILES

INNODB\_TABLESTATS

INNODB\_FOREIGN

INNODB\_COLUMNS

INNODB\_INDEXES

INNODB\_FIELDS

INNODB\_TABLESPACES

INNODB\_TABLESPACES\_BRIEF

INNODB\_FOREIGN\_COLS

INNODB\_TABLES

The table names are indicative of the type of data provided:

[**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) provides metadata about **InnoDB** tables.

[**INNODB\_COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-columns-table) provides metadata about **InnoDB** table columns.

[**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) provides metadata about **InnoDB** indexes.

[**INNODB\_FIELDS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-fields-table) provides metadata about the key columns (fields) of **InnoDB** indexes.

[**INNODB\_TABLESTATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table) provides a view of low-level status information about **InnoDB** tables that is derived from in-memory data structures.

[**INNODB\_DATAFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-datafiles-table) provides data file path information for **InnoDB** file-per-table and general tablespaces.

[**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) provides metadata about **InnoDB** file-per-table, general, and undo tablespaces.

[**INNODB\_TABLESPACES\_BRIEF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-brief-table) provides a subset of metadata about **InnoDB** tablespaces.

[**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) provides metadata about foreign keys defined on **InnoDB** tables.

[**INNODB\_FOREIGN\_COLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table) provides metadata about the columns of foreign keys that are defined on **InnoDB** tables.

**InnoDB** **INFORMATION\_SCHEMA** schema object tables can be joined together through fields such as **TABLE\_ID**, **INDEX\_ID**, and **SPACE**, allowing you to easily retrieve all available data for an object you want to study or monitor.

Refer to the **InnoDB** [INFORMATION\_SCHEMA](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#innodb-information-schema-tables) documentation for information about the columns of each table.

**Example 15.2 InnoDB INFORMATION\_SCHEMA Schema Object Tables**

This example uses a simple table (**t1**) with a single index (**i1**) to demonstrate the type of metadata found in the **InnoDB** **INFORMATION\_SCHEMA** schema object tables.

Create a test database and table **t1**:

mysql> **CREATE DATABASE test;**

mysql> **USE test;**

mysql> **CREATE TABLE t1 (**

**col1 INT,**

**col2 CHAR(10),**

**col3 VARCHAR(10))**

**ENGINE = InnoDB;**

mysql> **CREATE INDEX i1 ON t1(col1);**

After creating the table **t1**, query [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) to locate the metadata for **test/t1**:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLES WHERE NAME='test/t1' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 71

NAME: test/t1

FLAG: 1

N\_COLS: 6

SPACE: 57

ROW\_FORMAT: Compact

ZIP\_PAGE\_SIZE: 0

INSTANT\_COLS: 0

Table **t1** has a **TABLE\_ID** of 71. The **FLAG** field provides bit level information about table format and storage characteristics. There are six columns, three of which are hidden columns created by **InnoDB** (**DB\_ROW\_ID**, **DB\_TRX\_ID**, and **DB\_ROLL\_PTR**). The ID of the table's **SPACE** is 57 (a value of 0 would indicate that the table resides in the system tablespace). The **ROW\_FORMAT** is Compact. **ZIP\_PAGE\_SIZE** only applies to tables with a **Compressed** row format. **INSTANT\_COLS** shows number of columns in the table prior to adding the first instant column using **ALTER TABLE ... ADD COLUMN** with **ALGORITHM=INSTANT**.

Using the **TABLE\_ID** information from [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table), query the [**INNODB\_COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-columns-table) table for information about the table's columns.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_COLUMNS where TABLE\_ID = 71\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 71

NAME: col1

POS: 0

MTYPE: 6

PRTYPE: 1027

LEN: 4

HAS\_DEFAULT: 0

DEFAULT\_VALUE: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 71

NAME: col2

POS: 1

MTYPE: 2

PRTYPE: 524542

LEN: 10

HAS\_DEFAULT: 0

DEFAULT\_VALUE: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 71

NAME: col3

POS: 2

MTYPE: 1

PRTYPE: 524303

LEN: 10

HAS\_DEFAULT: 0

DEFAULT\_VALUE: NULL

In addition to the **TABLE\_ID** and column **NAME**, [**INNODB\_COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-columns-table) provides the ordinal position (**POS**) of each column (starting from 0 and incrementing sequentially), the column **MTYPE** or “main type” (6 = INT, 2 = CHAR, 1 = VARCHAR), the **PRTYPE** or “precise type” (a binary value with bits that represent the MySQL data type, character set code, and nullability), and the column length (**LEN**). The **HAS\_DEFAULT** and **DEFAULT\_VALUE** columns only apply to columns added instantly using **ALTER TABLE ... ADD COLUMN** with **ALGORITHM=INSTANT**.

Using the **TABLE\_ID** information from [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) once again, query [**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) for information about the indexes associated with table **t1**.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_INDEXES WHERE TABLE\_ID = 71 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INDEX\_ID: 111

NAME: GEN\_CLUST\_INDEX

TABLE\_ID: 71

TYPE: 1

N\_FIELDS: 0

PAGE\_NO: 3

SPACE: 57

MERGE\_THRESHOLD: 50

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INDEX\_ID: 112

NAME: i1

TABLE\_ID: 71

TYPE: 0

N\_FIELDS: 1

PAGE\_NO: 4

SPACE: 57

MERGE\_THRESHOLD: 50

[**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table) returns data for two indexes. The first index is **GEN\_CLUST\_INDEX**, which is a clustered index created by **InnoDB** if the table does not have a user-defined clustered index. The second index (**i1**) is the user-defined secondary index.

The **INDEX\_ID** is an identifier for the index that is unique across all databases in an instance. The **TABLE\_ID** identifies the table that the index is associated with. The index **TYPE** value indicates the type of index (1 = Clustered Index, 0 = Secondary index). The **N\_FILEDS** value is the number of fields that comprise the index. **PAGE\_NO** is the root page number of the index B-tree, and **SPACE** is the ID of the tablespace where the index resides. A nonzero value indicates that the index does not reside in the system tablespace. **MERGE\_THRESHOLD** defines a percentage threshold value for the amount of data in an index page. If the amount of data in an index page falls below the this value (the default is 50%) when a row is deleted or when a row is shortened by an update operation, **InnoDB** attempts to merge the index page with a neighboring index page.

Using the **INDEX\_ID** information from [**INNODB\_INDEXES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-indexes-table), query [**INNODB\_FIELDS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-fields-table) for information about the fields of index **i1**.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FIELDS where INDEX\_ID = 112 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INDEX\_ID: 112

NAME: col1

POS: 0

[**INNODB\_FIELDS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-fields-table) provides the **NAME** of the indexed field and its ordinal position within the index. If the index (i1) had been defined on multiple fields, [**INNODB\_FIELDS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-fields-table) would provide metadata for each of the indexed fields.

Using the **SPACE** information from [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table), query [**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) table for information about the table's tablespace.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLESPACES WHERE SPACE = 57 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 57

NAME: test/t1

FLAG: 16417

ROW\_FORMAT: Dynamic

PAGE\_SIZE: 16384

ZIP\_PAGE\_SIZE: 0

SPACE\_TYPE: Single

FS\_BLOCK\_SIZE: 4096

FILE\_SIZE: 114688

ALLOCATED\_SIZE: 98304

AUTOEXTEND\_SIZE: 0

SERVER\_VERSION: 8.0.23

SPACE\_VERSION: 1

ENCRYPTION: N

STATE: normal

In addition to the **SPACE** ID of the tablespace and the **NAME** of the associated table, [**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) provides tablespace **FLAG** data, which is bit level information about tablespace format and storage characteristics. Also provided are tablespace **ROW\_FORMAT**, **PAGE\_SIZE**, and several other tablespace metadata items.

Using the **SPACE** information from [**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table) once again, query [**INNODB\_DATAFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-datafiles-table) for the location of the tablespace data file.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_DATAFILES WHERE SPACE = 57 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SPACE: 57

PATH: ./test/t1.ibd

The datafile is located in the **test** directory under MySQL's **data** directory. If a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace were created in a location outside the MySQL data directory using the **DATA DIRECTORY** clause of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, the tablespace **PATH** would be a fully qualified directory path.

As a final step, insert a row into table **t1** (**TABLE\_ID = 71**) and view the data in the [**INNODB\_TABLESTATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table) table. The data in this table is used by the MySQL optimizer to calculate which index to use when querying an **InnoDB** table. This information is derived from in-memory data structures.

mysql> **INSERT INTO t1 VALUES(5, 'abc', 'def');**

Query OK, 1 row affected (0.06 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TABLESTATS where TABLE\_ID = 71 \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 71

NAME: test/t1

STATS\_INITIALIZED: Initialized

NUM\_ROWS: 1

CLUST\_INDEX\_SIZE: 1

OTHER\_INDEX\_SIZE: 0

MODIFIED\_COUNTER: 1

AUTOINC: 0

REF\_COUNT: 1

The **STATS\_INITIALIZED** field indicates whether or not statistics have been collected for the table. **NUM\_ROWS** is the current estimated number of rows in the table. The **CLUST\_INDEX\_SIZE** and **OTHER\_INDEX\_SIZE** fields report the number of pages on disk that store clustered and secondary indexes for the table, respectively. The **MODIFIED\_COUNTER** value shows the number of rows modified by DML operations and cascade operations from foreign keys. The **AUTOINC** value is the next number to be issued for any autoincrement-based operation. There are no autoincrement columns defined on table **t1**, so the value is 0. The **REF\_COUNT** value is a counter. When the counter reaches 0, it signifies that the table metadata can be evicted from the table cache.

**Example 15.3 Foreign Key INFORMATION\_SCHEMA Schema Object Tables**

The [**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) and [**INNODB\_FOREIGN\_COLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table) tables provide data about foreign key relationships. This example uses a parent table and child table with a foreign key relationship to demonstrate the data found in the [**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) and [**INNODB\_FOREIGN\_COLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table) tables.

Create the test database with parent and child tables:

mysql> **CREATE DATABASE test;**

mysql> **USE test;**

mysql> **CREATE TABLE parent (id INT NOT NULL,**

**PRIMARY KEY (id)) ENGINE=INNODB;**

mysql> **CREATE TABLE child (id INT, parent\_id INT,**

**INDEX par\_ind (parent\_id),**

**CONSTRAINT fk1**

**FOREIGN KEY (parent\_id) REFERENCES parent(id)**

**ON DELETE CASCADE) ENGINE=INNODB;**

After the parent and child tables are created, query [**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) and locate the foreign key data for the **test/child** and **test/parent** foreign key relationship:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FOREIGN \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ID: test/fk1

FOR\_NAME: test/child

REF\_NAME: test/parent

N\_COLS: 1

TYPE: 1

Metadata includes the foreign key **ID** (**fk1**), which is named for the **CONSTRAINT** that was defined on the child table. The **FOR\_NAME** is the name of the child table where the foreign key is defined. **REF\_NAME** is the name of the parent table (the “referenced” table). **N\_COLS** is the number of columns in the foreign key index. **TYPE** is a numerical value representing bit flags that provide additional information about the foreign key column. In this case, the **TYPE** value is 1, which indicates that the **ON DELETE CASCADE** option was specified for the foreign key. See the [**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) table definition for more information about **TYPE** values.

Using the foreign key **ID**, query [**INNODB\_FOREIGN\_COLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table) to view data about the columns of the foreign key.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FOREIGN\_COLS WHERE ID = 'test/fk1' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ID: test/fk1

FOR\_COL\_NAME: parent\_id

REF\_COL\_NAME: id

POS: 0

**FOR\_COL\_NAME** is the name of the foreign key column in the child table, and **REF\_COL\_NAME** is the name of the referenced column in the parent table. The **POS** value is the ordinal position of the key field within the foreign key index, starting at zero.

**Example 15.4 Joining InnoDB INFORMATION\_SCHEMA Schema Object Tables**

This example demonstrates joining three **InnoDB** **INFORMATION\_SCHEMA** schema object tables ([**INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table), [**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table), and [**INNODB\_TABLESTATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table)) to gather file format, row format, page size, and index size information about tables in the employees sample database.

The following table name aliases are used to shorten the query string:

[**INFORMATION\_SCHEMA.INNODB\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tables-table): a

[**INFORMATION\_SCHEMA.INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table): b

[**INFORMATION\_SCHEMA.INNODB\_TABLESTATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table): c

An [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) control flow function is used to account for compressed tables. If a table is compressed, the index size is calculated using **ZIP\_PAGE\_SIZE** rather than **PAGE\_SIZE**. **CLUST\_INDEX\_SIZE** and **OTHER\_INDEX\_SIZE**, which are reported in bytes, are divided by **1024\*1024** to provide index sizes in megabytes (MBs). MB values are rounded to zero decimal spaces using the [**ROUND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_round) function.

mysql> **SELECT a.NAME, a.ROW\_FORMAT,**

**@page\_size :=**

**IF(a.ROW\_FORMAT='Compressed',**

**b.ZIP\_PAGE\_SIZE, b.PAGE\_SIZE)**

**AS page\_size,**

**ROUND((@page\_size \* c.CLUST\_INDEX\_SIZE)**

**/(1024\*1024)) AS pk\_mb,**

**ROUND((@page\_size \* c.OTHER\_INDEX\_SIZE)**

**/(1024\*1024)) AS secidx\_mb**

**FROM INFORMATION\_SCHEMA.INNODB\_TABLES a**

**INNER JOIN INFORMATION\_SCHEMA.INNODB\_TABLESPACES b on a.NAME = b.NAME**

**INNER JOIN INFORMATION\_SCHEMA.INNODB\_TABLESTATS c on b.NAME = c.NAME**

**WHERE a.NAME LIKE 'employees/%'**

**ORDER BY a.NAME DESC;**

+------------------------+------------+-----------+-------+-----------+

| NAME | ROW\_FORMAT | page\_size | pk\_mb | secidx\_mb |

+------------------------+------------+-----------+-------+-----------+

| employees/titles | Dynamic | 16384 | 20 | 11 |

| employees/salaries | Dynamic | 16384 | 93 | 34 |

| employees/employees | Dynamic | 16384 | 15 | 0 |

| employees/dept\_manager | Dynamic | 16384 | 0 | 0 |

| employees/dept\_emp | Dynamic | 16384 | 12 | 10 |

| employees/departments | Dynamic | 16384 | 0 | 0 |

+------------------------+------------+-----------+-------+-----------+

### 15.15.4 InnoDB INFORMATION\_SCHEMA FULLTEXT Index Tables

The following tables provide metadata for **FULLTEXT** indexes:

mysql> **SHOW TABLES FROM INFORMATION\_SCHEMA LIKE 'INNODB\_FT%';**

+-------------------------------------------+

| Tables\_in\_INFORMATION\_SCHEMA (INNODB\_FT%) |

+-------------------------------------------+

| INNODB\_FT\_CONFIG |

| INNODB\_FT\_BEING\_DELETED |

| INNODB\_FT\_DELETED |

| INNODB\_FT\_DEFAULT\_STOPWORD |

| INNODB\_FT\_INDEX\_TABLE |

| INNODB\_FT\_INDEX\_CACHE |

+-------------------------------------------+

#### Table Overview

[**INNODB\_FT\_CONFIG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-config-table): Provides metadata about the **FULLTEXT** index and associated processing for an **InnoDB** table.

[**INNODB\_FT\_BEING\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-being-deleted-table): Provides a snapshot of the [**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table) table; it is used only during an [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) maintenance operation. When [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is run, the [**INNODB\_FT\_BEING\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-being-deleted-table) table is emptied, and **DOC\_ID** values are removed from the [**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table) table. Because the contents of [**INNODB\_FT\_BEING\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-being-deleted-table) typically have a short lifetime, this table has limited utility for monitoring or debugging. For information about running [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) on tables with **FULLTEXT** indexes, see [Section 12.10.6, “Fine-Tuning MySQL Full-Text Search”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-fine-tuning).

[**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table): Stores rows that are deleted from the **FULLTEXT** index for an **InnoDB** table. To avoid expensive index reorganization during DML operations for an **InnoDB** **FULLTEXT** index, the information about newly deleted words is stored separately, filtered out of search results when you do a text search, and removed from the main search index only when you issue an [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statement for the **InnoDB** table.

[**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table): Holds a list of [stopwords](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_stopword) that are used by default when creating a **FULLTEXT** index on **InnoDB** tables.

For information about the [**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table) table, see [Section 12.10.4, “Full-Text Stopwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-stopwords).

[**INNODB\_FT\_INDEX\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-table-table): Provides information about the inverted index used to process text searches against the **FULLTEXT** index of an **InnoDB** table.

[**INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table): Provides token information about newly inserted rows in a **FULLTEXT** index. To avoid expensive index reorganization during DML operations, the information about newly indexed words is stored separately, and combined with the main search index only when [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is run, when the server is shut down, or when the cache size exceeds a limit defined by the [**innodb\_ft\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_cache_size) or [**innodb\_ft\_total\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_total_cache_size) system variable.

**Note**

With the exception of the [**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table) table, these tables are empty initially. Before querying any of them, set the value of the [**innodb\_ft\_aux\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_aux_table) system variable to the name (including the database name) of the table that contains the **FULLTEXT** index (for example, **test/articles**).

**Example 15.5 InnoDB FULLTEXT Index INFORMATION\_SCHEMA Tables**

This example uses a table with a **FULLTEXT** index to demonstrate the data contained in the **FULLTEXT** index **INFORMATION\_SCHEMA** tables.

Create a table with a **FULLTEXT** index and insert some data:

mysql> **CREATE TABLE articles (**

**id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,**

**title VARCHAR(200),**

**body TEXT,**

**FULLTEXT (title,body)**

**) ENGINE=InnoDB;**

mysql> **INSERT INTO articles (title,body) VALUES**

**('MySQL Tutorial','DBMS stands for DataBase ...'),**

**('How To Use MySQL Well','After you went through a ...'),**

**('Optimizing MySQL','In this tutorial we show ...'),**

**('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),**

**('MySQL vs. YourSQL','In the following database comparison ...'),**

**('MySQL Security','When configured properly, MySQL ...');**

Set the [**innodb\_ft\_aux\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_aux_table) variable to the name of the table with the **FULLTEXT** index. If this variable is not set, the **InnoDB** **FULLTEXT** **INFORMATION\_SCHEMA** tables are empty, with the exception of [**INNODB\_FT\_DEFAULT\_STOPWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-default-stopword-table).

mysql> **SET GLOBAL innodb\_ft\_aux\_table = 'test/articles';**

Query the [**INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table) table, which shows information about newly inserted rows in a **FULLTEXT** index. To avoid expensive index reorganization during DML operations, data for newly inserted rows remains in the **FULLTEXT** index cache until [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is run (or until the server is shut down or cache limits are exceeded).

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE LIMIT 5;**

+------------+--------------+-------------+-----------+--------+----------+

| WORD | FIRST\_DOC\_ID | LAST\_DOC\_ID | DOC\_COUNT | DOC\_ID | POSITION |

+------------+--------------+-------------+-----------+--------+----------+

| 1001 | 5 | 5 | 1 | 5 | 0 |

| after | 3 | 3 | 1 | 3 | 22 |

| comparison | 6 | 6 | 1 | 6 | 44 |

| configured | 7 | 7 | 1 | 7 | 20 |

| database | 2 | 6 | 2 | 2 | 31 |

+------------+--------------+-------------+-----------+--------+----------+

Enable the [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) system variable and run [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) on the table that contains the **FULLTEXT** index. This operation flushes the contents of the **FULLTEXT** index cache to the main **FULLTEXT** index. [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) changes the way the [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statement operates on **InnoDB** tables, and is intended to be enabled temporarily, during maintenance operations on **InnoDB** tables with **FULLTEXT** indexes.

mysql> **SET GLOBAL innodb\_optimize\_fulltext\_only=ON;**

mysql> **OPTIMIZE TABLE articles;**

+---------------+----------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+---------------+----------+----------+----------+

| test.articles | optimize | status | OK |

+---------------+----------+----------+----------+

Query the [**INNODB\_FT\_INDEX\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-table-table) table to view information about data in the main **FULLTEXT** index, including information about the data that was just flushed from the **FULLTEXT** index cache.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_TABLE LIMIT 5;**

+------------+--------------+-------------+-----------+--------+----------+

| WORD | FIRST\_DOC\_ID | LAST\_DOC\_ID | DOC\_COUNT | DOC\_ID | POSITION |

+------------+--------------+-------------+-----------+--------+----------+

| 1001 | 5 | 5 | 1 | 5 | 0 |

| after | 3 | 3 | 1 | 3 | 22 |

| comparison | 6 | 6 | 1 | 6 | 44 |

| configured | 7 | 7 | 1 | 7 | 20 |

| database | 2 | 6 | 2 | 2 | 31 |

+------------+--------------+-------------+-----------+--------+----------+

The [**INNODB\_FT\_INDEX\_CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-index-cache-table) table is now empty since the [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operation flushed the **FULLTEXT** index cache.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_INDEX\_CACHE LIMIT 5;**

Empty set (0.00 sec)

Delete some records from the **test/articles** table.

mysql> **DELETE FROM test.articles WHERE id < 4;**

Query the [**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table) table. This table records rows that are deleted from the **FULLTEXT** index. To avoid expensive index reorganization during DML operations, information about newly deleted records is stored separately, filtered out of search results when you do a text search, and removed from the main search index when you run [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table).

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_DELETED;**

+--------+

| DOC\_ID |

+--------+

| 2 |

| 3 |

| 4 |

+--------+

Run [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) to remove the deleted records.

mysql> **OPTIMIZE TABLE articles;**

+---------------+----------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+---------------+----------+----------+----------+

| test.articles | optimize | status | OK |

+---------------+----------+----------+----------+

The [**INNODB\_FT\_DELETED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-deleted-table) table should now be empty.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_DELETED;**

Empty set (0.00 sec)

Query the [**INNODB\_FT\_CONFIG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-ft-config-table) table. This table contains metadata about the **FULLTEXT** index and related processing:

**optimize\_checkpoint\_limit**: The number of seconds after which an [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) run stops.

**synced\_doc\_id**: The next **DOC\_ID** to be issued.

**stopword\_table\_name**: The ***database/table*** name for a user-defined stopword table. The **VALUE** column is empty if there is no user-defined stopword table.

**use\_stopword**: Indicates whether a stopword table is used, which is defined when the **FULLTEXT** index is created.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FT\_CONFIG;**

+---------------------------+-------+

| KEY | VALUE |

+---------------------------+-------+

| optimize\_checkpoint\_limit | 180 |

| synced\_doc\_id | 8 |

| stopword\_table\_name | |

| use\_stopword | 1 |

+---------------------------+-------+

Disable [**innodb\_optimize\_fulltext\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only), since it is intended to be enabled only temporarily:

mysql> **SET GLOBAL innodb\_optimize\_fulltext\_only=OFF;**

### 15.15.5 InnoDB INFORMATION\_SCHEMA Buffer Pool Tables

The **InnoDB** **INFORMATION\_SCHEMA** buffer pool tables provide buffer pool status information and metadata about the pages within the **InnoDB** buffer pool.

The **InnoDB** **INFORMATION\_SCHEMA** buffer pool tables include those listed below:

mysql> **SHOW TABLES FROM INFORMATION\_SCHEMA LIKE 'INNODB\_BUFFER%';**

+-----------------------------------------------+

| Tables\_in\_INFORMATION\_SCHEMA (INNODB\_BUFFER%) |

+-----------------------------------------------+

| INNODB\_BUFFER\_PAGE\_LRU |

| INNODB\_BUFFER\_PAGE |

| INNODB\_BUFFER\_POOL\_STATS |

+-----------------------------------------------+

#### Table Overview

[**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table): Holds information about each page in the **InnoDB** buffer pool.

[**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table): Holds information about the pages in the **InnoDB** buffer pool, in particular how they are ordered in the LRU list that determines which pages to evict from the buffer pool when it becomes full. The [**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table) table has the same columns as the [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) table, except that the [**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table) table has an **LRU\_POSITION** column instead of a **BLOCK\_ID** column.

[**INNODB\_BUFFER\_POOL\_STATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-pool-stats-table): Provides buffer pool status information. Much of the same information is provided by [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output, or may be obtained using **InnoDB** buffer pool server status variables.

**Warning**

Querying the [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table) or [**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table) table can affect performance. Do not query these tables on a production system unless you are aware of the performance impact and have determined it to be acceptable. To avoid impacting performance on a production system, reproduce the issue you want to investigate and query buffer pool statistics on a test instance.

**Example 15.6 Querying System Data in the INNODB\_BUFFER\_PAGE Table**

This query provides an approximate count of pages that contain system data by excluding pages where the **TABLE\_NAME** value is either **NULL** or includes a slash **/** or period **.** in the table name, which indicates a user-defined table.

mysql> **SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME IS NULL OR (INSTR(TABLE\_NAME, '/') = 0 AND INSTR(TABLE\_NAME, '.') = 0);**

+----------+

| COUNT(\*) |

+----------+

| 1516 |

+----------+

This query returns the approximate number of pages that contain system data, the total number of buffer pool pages, and an approximate percentage of pages that contain system data.

mysql> **SELECT**

**(SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME IS NULL OR (INSTR(TABLE\_NAME, '/') = 0 AND INSTR(TABLE\_NAME, '.') = 0)**

**) AS system\_pages,**

**(**

**SELECT COUNT(\*)**

**FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**) AS total\_pages,**

**(**

**SELECT ROUND((system\_pages/total\_pages) \* 100)**

**) AS system\_page\_percentage;**

+--------------+-------------+------------------------+

| system\_pages | total\_pages | system\_page\_percentage |

+--------------+-------------+------------------------+

| 295 | 8192 | 4 |

+--------------+-------------+------------------------+

The type of system data in the buffer pool can be determined by querying the **PAGE\_TYPE** value. For example, the following query returns eight distinct **PAGE\_TYPE** values among the pages that contain system data:

mysql> **SELECT DISTINCT PAGE\_TYPE FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME IS NULL OR (INSTR(TABLE\_NAME, '/') = 0 AND INSTR(TABLE\_NAME, '.') = 0);**

+-------------------+

| PAGE\_TYPE |

+-------------------+

| SYSTEM |

| IBUF\_BITMAP |

| UNKNOWN |

| FILE\_SPACE\_HEADER |

| INODE |

| UNDO\_LOG |

| ALLOCATED |

+-------------------+

**Example 15.7 Querying User Data in the INNODB\_BUFFER\_PAGE Table**

This query provides an approximate count of pages containing user data by counting pages where the **TABLE\_NAME** value is **NOT NULL** and **NOT LIKE '%INNODB\_TABLES%'**.

mysql> **SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME IS NOT NULL AND TABLE\_NAME NOT LIKE '%INNODB\_TABLES%';**

+----------+

| COUNT(\*) |

+----------+

| 7897 |

+----------+

This query returns the approximate number of pages that contain user data, the total number of buffer pool pages, and an approximate percentage of pages that contain user data.

mysql> **SELECT**

**(SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

WHERE TABLE\_NAME IS NOT NULL AND (INSTR(TABLE\_NAME, '/') > 0 OR INSTR(TABLE\_NAME, '.') > 0)

**) AS user\_pages,**

**(**

**SELECT COUNT(\*)**

**FROM information\_schema.INNODB\_BUFFER\_PAGE**

**) AS total\_pages,**

**(**

**SELECT ROUND((user\_pages/total\_pages) \* 100)**

**) AS user\_page\_percentage;**

+------------+-------------+----------------------+

| user\_pages | total\_pages | user\_page\_percentage |

+------------+-------------+----------------------+

| 7897 | 8192 | 96 |

+------------+-------------+----------------------+

This query identifies user-defined tables with pages in the buffer pool:

mysql> **SELECT DISTINCT TABLE\_NAME FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME IS NOT NULL AND (INSTR(TABLE\_NAME, '/') > 0 OR INSTR(TABLE\_NAME, '.') > 0)**

**AND TABLE\_NAME NOT LIKE '`mysql`.`innodb\_%';**

+-------------------------+

| TABLE\_NAME |

+-------------------------+

| `employees`.`salaries` |

| `employees`.`employees` |

+-------------------------+

**Example 15.8 Querying Index Data in the INNODB\_BUFFER\_PAGE Table**

For information about index pages, query the **INDEX\_NAME** column using the name of the index. For example, the following query returns the number of pages and total data size of pages for the **emp\_no** index that is defined on the **employees.salaries** table:

mysql> **SELECT INDEX\_NAME, COUNT(\*) AS Pages,**

**ROUND(SUM(IF(COMPRESSED\_SIZE = 0, @@GLOBAL.innodb\_page\_size, COMPRESSED\_SIZE))/1024/1024)**

**AS 'Total Data (MB)'**

**FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE INDEX\_NAME='emp\_no' AND TABLE\_NAME = '`employees`.`salaries`';**

+------------+-------+-----------------+

| INDEX\_NAME | Pages | Total Data (MB) |

+------------+-------+-----------------+

| emp\_no | 1609 | 25 |

+------------+-------+-----------------+

This query returns the number of pages and total data size of pages for all indexes defined on the **employees.salaries** table:

mysql> **SELECT INDEX\_NAME, COUNT(\*) AS Pages,**

**ROUND(SUM(IF(COMPRESSED\_SIZE = 0, @@GLOBAL.innodb\_page\_size, COMPRESSED\_SIZE))/1024/1024)**

**AS 'Total Data (MB)'**

**FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE**

**WHERE TABLE\_NAME = '`employees`.`salaries`'**

**GROUP BY INDEX\_NAME;**

+------------+-------+-----------------+

| INDEX\_NAME | Pages | Total Data (MB) |

+------------+-------+-----------------+

| emp\_no | 1608 | 25 |

| PRIMARY | 6086 | 95 |

+------------+-------+-----------------+

**Example 15.9 Querying LRU\_POSITION Data in the INNODB\_BUFFER\_PAGE\_LRU Table**

The [**INNODB\_BUFFER\_PAGE\_LRU**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-lru-table) table holds information about the pages in the **InnoDB** buffer pool, in particular how they are ordered that determines which pages to evict from the buffer pool when it becomes full. The definition for this page is the same as for [**INNODB\_BUFFER\_PAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-page-table), except this table has an **LRU\_POSITION** column instead of a **BLOCK\_ID** column.

This query counts the number of positions at a specific location in the LRU list occupied by pages of the **employees.employees** table.

mysql> **SELECT COUNT(LRU\_POSITION) FROM INFORMATION\_SCHEMA.INNODB\_BUFFER\_PAGE\_LRU**

**WHERE TABLE\_NAME='`employees`.`employees`' AND LRU\_POSITION < 3072;**

+---------------------+

| COUNT(LRU\_POSITION) |

+---------------------+

| 548 |

+---------------------+

**Example 15.10 Querying the INNODB\_BUFFER\_POOL\_STATS Table**

The [**INNODB\_BUFFER\_POOL\_STATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-buffer-pool-stats-table) table provides information similar to [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) and **InnoDB** buffer pool status variables.

mysql> **SELECT \* FROM information\_schema.INNODB\_BUFFER\_POOL\_STATS \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

POOL\_ID: 0

POOL\_SIZE: 8192

FREE\_BUFFERS: 1

DATABASE\_PAGES: 8173

OLD\_DATABASE\_PAGES: 3014

MODIFIED\_DATABASE\_PAGES: 0

PENDING\_DECOMPRESS: 0

PENDING\_READS: 0

PENDING\_FLUSH\_LRU: 0

PENDING\_FLUSH\_LIST: 0

PAGES\_MADE\_YOUNG: 15907

PAGES\_NOT\_MADE\_YOUNG: 3803101

PAGES\_MADE\_YOUNG\_RATE: 0

PAGES\_MADE\_NOT\_YOUNG\_RATE: 0

NUMBER\_PAGES\_READ: 3270

NUMBER\_PAGES\_CREATED: 13176

NUMBER\_PAGES\_WRITTEN: 15109

PAGES\_READ\_RATE: 0

PAGES\_CREATE\_RATE: 0

PAGES\_WRITTEN\_RATE: 0

NUMBER\_PAGES\_GET: 33069332

HIT\_RATE: 0

YOUNG\_MAKE\_PER\_THOUSAND\_GETS: 0

NOT\_YOUNG\_MAKE\_PER\_THOUSAND\_GETS: 0

NUMBER\_PAGES\_READ\_AHEAD: 2713

NUMBER\_READ\_AHEAD\_EVICTED: 0

READ\_AHEAD\_RATE: 0

READ\_AHEAD\_EVICTED\_RATE: 0

LRU\_IO\_TOTAL: 0

LRU\_IO\_CURRENT: 0

UNCOMPRESS\_TOTAL: 0

UNCOMPRESS\_CURRENT: 0

For comparison, [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output and **InnoDB** buffer pool status variable output is shown below, based on the same data set.

For more information about [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output, see [Section 15.17.3, “InnoDB Standard Monitor and Lock Monitor Output”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor).

mysql> **SHOW ENGINE INNODB STATUS \G**

...

----------------------

BUFFER POOL AND MEMORY

----------------------

Total large memory allocated 137428992

Dictionary memory allocated 579084

Buffer pool size 8192

Free buffers 1

Database pages 8173

Old database pages 3014

Modified db pages 0

Pending reads 0

Pending writes: LRU 0, flush list 0, single page 0

Pages made young 15907, not young 3803101

0.00 youngs/s, 0.00 non-youngs/s

Pages read 3270, created 13176, written 15109

0.00 reads/s, 0.00 creates/s, 0.00 writes/s

No buffer pool page gets since the last printout

Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s

LRU len: 8173, unzip\_LRU len: 0

I/O sum[0]:cur[0], unzip sum[0]:cur[0]

...

For status variable descriptions, see [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

mysql> **SHOW STATUS LIKE 'Innodb\_buffer%';**

+---------------------------------------+-------------+

| Variable\_name | Value |

+---------------------------------------+-------------+

| Innodb\_buffer\_pool\_dump\_status | not started |

| Innodb\_buffer\_pool\_load\_status | not started |

| Innodb\_buffer\_pool\_resize\_status | not started |

| Innodb\_buffer\_pool\_pages\_data | 8173 |

| Innodb\_buffer\_pool\_bytes\_data | 133906432 |

| Innodb\_buffer\_pool\_pages\_dirty | 0 |

| Innodb\_buffer\_pool\_bytes\_dirty | 0 |

| Innodb\_buffer\_pool\_pages\_flushed | 15109 |

| Innodb\_buffer\_pool\_pages\_free | 1 |

| Innodb\_buffer\_pool\_pages\_misc | 18 |

| Innodb\_buffer\_pool\_pages\_total | 8192 |

| Innodb\_buffer\_pool\_read\_ahead\_rnd | 0 |

| Innodb\_buffer\_pool\_read\_ahead | 2713 |

| Innodb\_buffer\_pool\_read\_ahead\_evicted | 0 |

| Innodb\_buffer\_pool\_read\_requests | 33069332 |

| Innodb\_buffer\_pool\_reads | 558 |

| Innodb\_buffer\_pool\_wait\_free | 0 |

| Innodb\_buffer\_pool\_write\_requests | 11985961 |

+---------------------------------------+-------------+

### 15.15.6 InnoDB INFORMATION\_SCHEMA Metrics Table

The [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table provides information about **InnoDB** performance and resource-related counters.

[**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table columns are shown below. For column descriptions, see [Section 26.4.23, “The INFORMATION\_SCHEMA INNODB\_METRICS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table).

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts" \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 46273

MAX\_COUNT: 46273

MIN\_COUNT: NULL

AVG\_COUNT: 492.2659574468085

COUNT\_RESET: 46273

MAX\_COUNT\_RESET: 46273

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: NULL

TIME\_ENABLED: 2014-11-28 16:07:53

TIME\_DISABLED: NULL

TIME\_ELAPSED: 94

TIME\_RESET: NULL

STATUS: enabled

TYPE: status\_counter

COMMENT: Number of rows inserted

#### Enabling, Disabling, and Resetting Counters

You can enable, disable, and reset counters using the following variables:

[**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable): Enables counters.

SET GLOBAL innodb\_monitor\_enable = [counter-name|module\_name|pattern|all];

[**innodb\_monitor\_disable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable): Disables counters.

SET GLOBAL innodb\_monitor\_disable = [counter-name|module\_name|pattern|all];

[**innodb\_monitor\_reset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset): Resets counter values to zero.

SET GLOBAL innodb\_monitor\_reset = [counter-name|module\_name|pattern|all];

[**innodb\_monitor\_reset\_all**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset_all): Resets all counter values. A counter must be disabled before using [**innodb\_monitor\_reset\_all**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset_all).

SET GLOBAL innodb\_monitor\_reset\_all = [counter-name|module\_name|pattern|all];

Counters and counter modules can also be enabled at startup using the MySQL server configuration file. For example, to enable the **log** module, **metadata\_table\_handles\_opened** and **metadata\_table\_handles\_closed** counters, enter the following line in the **[mysqld]** section of the MySQL server configuration file.

[mysqld]

innodb\_monitor\_enable = module\_recovery,metadata\_table\_handles\_opened,metadata\_table\_handles\_closed

When enabling multiple counters or modules in a configuration file, specify the [**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) variable followed by counter and module names separated by a comma, as shown above. Only the [**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) variable can be used in a configuration file. The [**innodb\_monitor\_disable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable) and [**innodb\_monitor\_reset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset) variables are supported on the command line only.

**Note**

Because each counter adds a degree of runtime overhead, use counters conservatively on production servers to diagnose specific issues or monitor specific functionality. A test or development server is recommended for more extensive use of counters.

#### Counters

The list of available counters is subject to change. Query the [**INFORMATION\_SCHEMA.INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table for counters available in your MySQL server version.

The counters enabled by default correspond to those shown in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output. Counters shown in [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output are always enabled at a system level but can be disable for the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table. Counter status is not persistent. Unless configured otherwise, counters revert to their default enabled or disabled status when the server is restarted.

If you run programs that would be affected by the addition or removal of counters, it is recommended that you review the releases notes and query the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table to identify those changes as part of your upgrade process.

mysql> **SELECT name, subsystem, status FROM INFORMATION\_SCHEMA.INNODB\_METRICS ORDER BY NAME;**

+------------------------------------------+---------------------+----------+

| name | subsystem | status |

+------------------------------------------+---------------------+----------+

| adaptive\_hash\_pages\_added | adaptive\_hash\_index | disabled |

| adaptive\_hash\_pages\_removed | adaptive\_hash\_index | disabled |

| adaptive\_hash\_rows\_added | adaptive\_hash\_index | disabled |

| adaptive\_hash\_rows\_deleted\_no\_hash\_entry | adaptive\_hash\_index | disabled |

| adaptive\_hash\_rows\_removed | adaptive\_hash\_index | disabled |

| adaptive\_hash\_rows\_updated | adaptive\_hash\_index | disabled |

| adaptive\_hash\_searches | adaptive\_hash\_index | enabled |

| adaptive\_hash\_searches\_btree | adaptive\_hash\_index | enabled |

| buffer\_data\_reads | buffer | enabled |

| buffer\_data\_written | buffer | enabled |

| buffer\_flush\_adaptive | buffer | disabled |

| buffer\_flush\_adaptive\_avg\_pass | buffer | disabled |

| buffer\_flush\_adaptive\_avg\_time\_est | buffer | disabled |

| buffer\_flush\_adaptive\_avg\_time\_slot | buffer | disabled |

| buffer\_flush\_adaptive\_avg\_time\_thread | buffer | disabled |

| buffer\_flush\_adaptive\_pages | buffer | disabled |

| buffer\_flush\_adaptive\_total\_pages | buffer | disabled |

| buffer\_flush\_avg\_page\_rate | buffer | disabled |

| buffer\_flush\_avg\_pass | buffer | disabled |

| buffer\_flush\_avg\_time | buffer | disabled |

| buffer\_flush\_background | buffer | disabled |

| buffer\_flush\_background\_pages | buffer | disabled |

| buffer\_flush\_background\_total\_pages | buffer | disabled |

| buffer\_flush\_batches | buffer | disabled |

| buffer\_flush\_batch\_num\_scan | buffer | disabled |

| buffer\_flush\_batch\_pages | buffer | disabled |

| buffer\_flush\_batch\_scanned | buffer | disabled |

| buffer\_flush\_batch\_scanned\_per\_call | buffer | disabled |

| buffer\_flush\_batch\_total\_pages | buffer | disabled |

| buffer\_flush\_lsn\_avg\_rate | buffer | disabled |

| buffer\_flush\_neighbor | buffer | disabled |

| buffer\_flush\_neighbor\_pages | buffer | disabled |

| buffer\_flush\_neighbor\_total\_pages | buffer | disabled |

| buffer\_flush\_n\_to\_flush\_by\_age | buffer | disabled |

| buffer\_flush\_n\_to\_flush\_requested | buffer | disabled |

| buffer\_flush\_pct\_for\_dirty | buffer | disabled |

| buffer\_flush\_pct\_for\_lsn | buffer | disabled |

| buffer\_flush\_sync | buffer | disabled |

| buffer\_flush\_sync\_pages | buffer | disabled |

| buffer\_flush\_sync\_total\_pages | buffer | disabled |

| buffer\_flush\_sync\_waits | buffer | disabled |

| buffer\_LRU\_batches\_evict | buffer | disabled |

| buffer\_LRU\_batches\_flush | buffer | disabled |

| buffer\_LRU\_batch\_evict\_pages | buffer | disabled |

| buffer\_LRU\_batch\_evict\_total\_pages | buffer | disabled |

| buffer\_LRU\_batch\_flush\_avg\_pass | buffer | disabled |

| buffer\_LRU\_batch\_flush\_avg\_time\_est | buffer | disabled |

| buffer\_LRU\_batch\_flush\_avg\_time\_slot | buffer | disabled |

| buffer\_LRU\_batch\_flush\_avg\_time\_thread | buffer | disabled |

| buffer\_LRU\_batch\_flush\_pages | buffer | disabled |

| buffer\_LRU\_batch\_flush\_total\_pages | buffer | disabled |

| buffer\_LRU\_batch\_num\_scan | buffer | disabled |

| buffer\_LRU\_batch\_scanned | buffer | disabled |

| buffer\_LRU\_batch\_scanned\_per\_call | buffer | disabled |

| buffer\_LRU\_get\_free\_loops | buffer | disabled |

| buffer\_LRU\_get\_free\_search | Buffer | disabled |

| buffer\_LRU\_get\_free\_waits | buffer | disabled |

| buffer\_LRU\_search\_num\_scan | buffer | disabled |

| buffer\_LRU\_search\_scanned | buffer | disabled |

| buffer\_LRU\_search\_scanned\_per\_call | buffer | disabled |

| buffer\_LRU\_single\_flush\_failure\_count | Buffer | disabled |

| buffer\_LRU\_single\_flush\_num\_scan | buffer | disabled |

| buffer\_LRU\_single\_flush\_scanned | buffer | disabled |

| buffer\_LRU\_single\_flush\_scanned\_per\_call | buffer | disabled |

| buffer\_LRU\_unzip\_search\_num\_scan | buffer | disabled |

| buffer\_LRU\_unzip\_search\_scanned | buffer | disabled |

| buffer\_LRU\_unzip\_search\_scanned\_per\_call | buffer | disabled |

| buffer\_pages\_created | buffer | enabled |

| buffer\_pages\_read | buffer | enabled |

| buffer\_pages\_written | buffer | enabled |

| buffer\_page\_read\_blob | buffer\_page\_io | disabled |

| buffer\_page\_read\_fsp\_hdr | buffer\_page\_io | disabled |

| buffer\_page\_read\_ibuf\_bitmap | buffer\_page\_io | disabled |

| buffer\_page\_read\_ibuf\_free\_list | buffer\_page\_io | disabled |

| buffer\_page\_read\_index\_ibuf\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_read\_index\_ibuf\_non\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_read\_index\_inode | buffer\_page\_io | disabled |

| buffer\_page\_read\_index\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_read\_index\_non\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_read\_other | buffer\_page\_io | disabled |

| buffer\_page\_read\_system\_page | buffer\_page\_io | disabled |

| buffer\_page\_read\_trx\_system | buffer\_page\_io | disabled |

| buffer\_page\_read\_undo\_log | buffer\_page\_io | disabled |

| buffer\_page\_read\_xdes | buffer\_page\_io | disabled |

| buffer\_page\_read\_zblob | buffer\_page\_io | disabled |

| buffer\_page\_read\_zblob2 | buffer\_page\_io | disabled |

| buffer\_page\_written\_blob | buffer\_page\_io | disabled |

| buffer\_page\_written\_fsp\_hdr | buffer\_page\_io | disabled |

| buffer\_page\_written\_ibuf\_bitmap | buffer\_page\_io | disabled |

| buffer\_page\_written\_ibuf\_free\_list | buffer\_page\_io | disabled |

| buffer\_page\_written\_index\_ibuf\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_written\_index\_ibuf\_non\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_written\_index\_inode | buffer\_page\_io | disabled |

| buffer\_page\_written\_index\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_written\_index\_non\_leaf | buffer\_page\_io | disabled |

| buffer\_page\_written\_other | buffer\_page\_io | disabled |

| buffer\_page\_written\_system\_page | buffer\_page\_io | disabled |

| buffer\_page\_written\_trx\_system | buffer\_page\_io | disabled |

| buffer\_page\_written\_undo\_log | buffer\_page\_io | disabled |

| buffer\_page\_written\_xdes | buffer\_page\_io | disabled |

| buffer\_page\_written\_zblob | buffer\_page\_io | disabled |

| buffer\_page\_written\_zblob2 | buffer\_page\_io | disabled |

| buffer\_pool\_bytes\_data | buffer | enabled |

| buffer\_pool\_bytes\_dirty | buffer | enabled |

| buffer\_pool\_pages\_data | buffer | enabled |

| buffer\_pool\_pages\_dirty | buffer | enabled |

| buffer\_pool\_pages\_free | buffer | enabled |

| buffer\_pool\_pages\_misc | buffer | enabled |

| buffer\_pool\_pages\_total | buffer | enabled |

| buffer\_pool\_reads | buffer | enabled |

| buffer\_pool\_read\_ahead | buffer | enabled |

| buffer\_pool\_read\_ahead\_evicted | buffer | enabled |

| buffer\_pool\_read\_requests | buffer | enabled |

| buffer\_pool\_size | server | enabled |

| buffer\_pool\_wait\_free | buffer | enabled |

| buffer\_pool\_write\_requests | buffer | enabled |

| compression\_pad\_decrements | compression | disabled |

| compression\_pad\_increments | compression | disabled |

| compress\_pages\_compressed | compression | disabled |

| compress\_pages\_decompressed | compression | disabled |

| ddl\_background\_drop\_indexes | ddl | disabled |

| ddl\_background\_drop\_tables | ddl | disabled |

| ddl\_log\_file\_alter\_table | ddl | disabled |

| ddl\_online\_create\_index | ddl | disabled |

| ddl\_pending\_alter\_table | ddl | disabled |

| ddl\_sort\_file\_alter\_table | ddl | disabled |

| dml\_deletes | dml | enabled |

| dml\_inserts | dml | enabled |

| dml\_reads | dml | disabled |

| dml\_updates | dml | enabled |

| file\_num\_open\_files | file\_system | enabled |

| ibuf\_merges | change\_buffer | enabled |

| ibuf\_merges\_delete | change\_buffer | enabled |

| ibuf\_merges\_delete\_mark | change\_buffer | enabled |

| ibuf\_merges\_discard\_delete | change\_buffer | enabled |

| ibuf\_merges\_discard\_delete\_mark | change\_buffer | enabled |

| ibuf\_merges\_discard\_insert | change\_buffer | enabled |

| ibuf\_merges\_insert | change\_buffer | enabled |

| ibuf\_size | change\_buffer | enabled |

| icp\_attempts | icp | disabled |

| icp\_match | icp | disabled |

| icp\_no\_match | icp | disabled |

| icp\_out\_of\_range | icp | disabled |

| index\_page\_discards | index | disabled |

| index\_page\_merge\_attempts | index | disabled |

| index\_page\_merge\_successful | index | disabled |

| index\_page\_reorg\_attempts | index | disabled |

| index\_page\_reorg\_successful | index | disabled |

| index\_page\_splits | index | disabled |

| innodb\_activity\_count | server | enabled |

| innodb\_background\_drop\_table\_usec | server | disabled |

| innodb\_checkpoint\_usec | server | disabled |

| innodb\_dblwr\_pages\_written | server | enabled |

| innodb\_dblwr\_writes | server | enabled |

| innodb\_dict\_lru\_count | server | disabled |

| innodb\_dict\_lru\_usec | server | disabled |

| innodb\_ibuf\_merge\_usec | server | disabled |

| innodb\_log\_flush\_usec | server | disabled |

| innodb\_master\_active\_loops | server | disabled |

| innodb\_master\_idle\_loops | server | disabled |

| innodb\_master\_purge\_usec | server | disabled |

| innodb\_master\_thread\_sleeps | server | disabled |

| innodb\_mem\_validate\_usec | server | disabled |

| innodb\_page\_size | server | enabled |

| innodb\_rwlock\_sx\_os\_waits | server | enabled |

| innodb\_rwlock\_sx\_spin\_rounds | server | enabled |

| innodb\_rwlock\_sx\_spin\_waits | server | enabled |

| innodb\_rwlock\_s\_os\_waits | server | enabled |

| innodb\_rwlock\_s\_spin\_rounds | server | enabled |

| innodb\_rwlock\_s\_spin\_waits | server | enabled |

| innodb\_rwlock\_x\_os\_waits | server | enabled |

| innodb\_rwlock\_x\_spin\_rounds | server | enabled |

| innodb\_rwlock\_x\_spin\_waits | server | enabled |

| lock\_deadlocks | lock | enabled |

| lock\_rec\_locks | lock | disabled |

| lock\_rec\_lock\_created | lock | disabled |

| lock\_rec\_lock\_removed | lock | disabled |

| lock\_rec\_lock\_requests | lock | disabled |

| lock\_rec\_lock\_waits | lock | disabled |

| lock\_row\_lock\_current\_waits | lock | enabled |

| lock\_row\_lock\_time | lock | enabled |

| lock\_row\_lock\_time\_avg | lock | enabled |

| lock\_row\_lock\_time\_max | lock | enabled |

| lock\_row\_lock\_waits | lock | enabled |

| lock\_table\_locks | lock | disabled |

| lock\_table\_lock\_created | lock | disabled |

| lock\_table\_lock\_removed | lock | disabled |

| lock\_table\_lock\_waits | lock | disabled |

| lock\_timeouts | lock | enabled |

| log\_checkpoints | recovery | disabled |

| log\_lsn\_buf\_pool\_oldest | recovery | disabled |

| log\_lsn\_checkpoint\_age | recovery | disabled |

| log\_lsn\_current | recovery | disabled |

| log\_lsn\_last\_checkpoint | recovery | disabled |

| log\_lsn\_last\_flush | recovery | disabled |

| log\_max\_modified\_age\_async | recovery | disabled |

| log\_max\_modified\_age\_sync | recovery | disabled |

| log\_num\_log\_io | recovery | disabled |

| log\_padded | recovery | enabled |

| log\_pending\_checkpoint\_writes | recovery | disabled |

| log\_pending\_log\_flushes | recovery | disabled |

| log\_waits | recovery | enabled |

| log\_writes | recovery | enabled |

| log\_write\_requests | recovery | enabled |

| metadata\_table\_handles\_closed | metadata | disabled |

| metadata\_table\_handles\_opened | metadata | disabled |

| metadata\_table\_reference\_count | metadata | disabled |

| os\_data\_fsyncs | os | enabled |

| os\_data\_reads | os | enabled |

| os\_data\_writes | os | enabled |

| os\_log\_bytes\_written | os | enabled |

| os\_log\_fsyncs | os | enabled |

| os\_log\_pending\_fsyncs | os | enabled |

| os\_log\_pending\_writes | os | enabled |

| os\_pending\_reads | os | disabled |

| os\_pending\_writes | os | disabled |

| purge\_del\_mark\_records | purge | disabled |

| purge\_dml\_delay\_usec | purge | disabled |

| purge\_invoked | purge | disabled |

| purge\_resume\_count | purge | disabled |

| purge\_stop\_count | purge | disabled |

| purge\_undo\_log\_pages | purge | disabled |

| purge\_upd\_exist\_or\_extern\_records | purge | disabled |

| trx\_active\_transactions | transaction | disabled |

| trx\_commits\_insert\_update | transaction | disabled |

| trx\_nl\_ro\_commits | transaction | disabled |

| trx\_rollbacks | transaction | disabled |

| trx\_rollbacks\_savepoint | transaction | disabled |

| trx\_rollback\_active | transaction | disabled |

| trx\_ro\_commits | transaction | disabled |

| trx\_rseg\_current\_size | transaction | disabled |

| trx\_rseg\_history\_len | transaction | enabled |

| trx\_rw\_commits | transaction | disabled |

| trx\_undo\_slots\_cached | transaction | disabled |

| trx\_undo\_slots\_used | transaction | disabled |

+------------------------------------------+---------------------+----------+

235 rows in set (0.01 sec)

#### Counter Modules

Each counter is associated with a particular module. Module names can be used to enable, disable, or reset all counters for a particular subsystem. For example, use **module\_dml** to enable all counters associated with the **dml** subsystem.

mysql> **SET GLOBAL innodb\_monitor\_enable = module\_dml;**

mysql> **SELECT name, subsystem, status FROM INFORMATION\_SCHEMA.INNODB\_METRICS**

**WHERE subsystem ='dml';**

+-------------+-----------+---------+

| name | subsystem | status |

+-------------+-----------+---------+

| dml\_reads | dml | enabled |

| dml\_inserts | dml | enabled |

| dml\_deletes | dml | enabled |

| dml\_updates | dml | enabled |

+-------------+-----------+---------+

Module names can be used with [**innodb\_monitor\_enable**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable) and related variables.

Module names and corresponding **SUBSYSTEM** names are listed below.

**module\_adaptive\_hash** (subsystem = **adaptive\_hash\_index**)

**module\_buffer** (subsystem = **buffer**)

**module\_buffer\_page** (subsystem = **buffer\_page\_io**)

**module\_compress** (subsystem = **compression**)

**module\_ddl** (subsystem = **ddl**)

**module\_dml** (subsystem = **dml**)

**module\_file** (subsystem = **file\_system**)

**module\_ibuf\_system** (subsystem = **change\_buffer**)

**module\_icp** (subsystem = **icp**)

**module\_index** (subsystem = **index**)

**module\_innodb** (subsystem = **innodb**)

**module\_lock** (subsystem = **lock**)

**module\_log** (subsystem = **recovery**)

**module\_metadata** (subsystem = **metadata**)

**module\_os** (subsystem = **os**)

**module\_purge** (subsystem = **purge**)

**module\_trx** (subsystem = **transaction**)

**module\_undo** (subsystem = **undo**)

**Example 15.11 Working with INNODB\_METRICS Table Counters**

This example demonstrates enabling, disabling, and resetting a counter, and querying counter data in the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table.

Create a simple **InnoDB** table:

mysql> **USE test;**

Database changed

mysql> **CREATE TABLE t1 (c1 INT) ENGINE=INNODB;**

Query OK, 0 rows affected (0.02 sec)

Enable the **dml\_inserts** counter.

mysql> **SET GLOBAL innodb\_monitor\_enable = dml\_inserts;**

Query OK, 0 rows affected (0.01 sec)

A description of the **dml\_inserts** counter can be found in the **COMMENT** column of the **INNODB\_METRICS** table:

mysql> **SELECT NAME, COMMENT FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts";**

+-------------+-------------------------+

| NAME | COMMENT |

+-------------+-------------------------+

| dml\_inserts | Number of rows inserted |

+-------------+-------------------------+

Query the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table for the **dml\_inserts** counter data. Because no DML operations have been performed, the counter values are zero or NULL. The **TIME\_ENABLED** and **TIME\_ELAPSED** values indicate when the counter was last enabled and how many seconds have elapsed since that time.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts" \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 0

MAX\_COUNT: 0

MIN\_COUNT: NULL

AVG\_COUNT: 0

COUNT\_RESET: 0

MAX\_COUNT\_RESET: 0

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: NULL

TIME\_ENABLED: 2014-12-04 14:18:28

TIME\_DISABLED: NULL

TIME\_ELAPSED: 28

TIME\_RESET: NULL

STATUS: enabled

TYPE: status\_counter

COMMENT: Number of rows inserted

Insert three rows of data into the table.

mysql> **INSERT INTO t1 values(1);**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO t1 values(2);**

Query OK, 1 row affected (0.00 sec)

mysql> **INSERT INTO t1 values(3);**

Query OK, 1 row affected (0.00 sec)

Query the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table again for the **dml\_inserts** counter data. A number of counter values have now incremented including **COUNT**, **MAX\_COUNT**, **AVG\_COUNT**, and **COUNT\_RESET**. Refer to the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table definition for descriptions of these values.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 3

MAX\_COUNT: 3

MIN\_COUNT: NULL

AVG\_COUNT: 0.046153846153846156

COUNT\_RESET: 3

MAX\_COUNT\_RESET: 3

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: NULL

TIME\_ENABLED: 2014-12-04 14:18:28

TIME\_DISABLED: NULL

TIME\_ELAPSED: 65

TIME\_RESET: NULL

STATUS: enabled

TYPE: status\_counter

COMMENT: Number of rows inserted

Reset the **dml\_inserts** counter and query the [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) table again for the **dml\_inserts** counter data. The **%\_RESET** values that were reported previously, such as **COUNT\_RESET** and **MAX\_RESET**, are set back to zero. Values such as **COUNT**, **MAX\_COUNT**, and **AVG\_COUNT**, which cumulatively collect data from the time the counter is enabled, are unaffected by the reset.

mysql> **SET GLOBAL innodb\_monitor\_reset = dml\_inserts;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 3

MAX\_COUNT: 3

MIN\_COUNT: NULL

AVG\_COUNT: 0.03529411764705882

COUNT\_RESET: 0

MAX\_COUNT\_RESET: 0

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: 0

TIME\_ENABLED: 2014-12-04 14:18:28

TIME\_DISABLED: NULL

TIME\_ELAPSED: 85

TIME\_RESET: 2014-12-04 14:19:44

STATUS: enabled

TYPE: status\_counter

COMMENT: Number of rows inserted

To reset all counter values, you must first disable the counter. Disabling the counter sets the **STATUS** value to **disabled**.

mysql> **SET GLOBAL innodb\_monitor\_disable = dml\_inserts;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 3

MAX\_COUNT: 3

MIN\_COUNT: NULL

AVG\_COUNT: 0.030612244897959183

COUNT\_RESET: 0

MAX\_COUNT\_RESET: 0

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: 0

TIME\_ENABLED: 2014-12-04 14:18:28

TIME\_DISABLED: 2014-12-04 14:20:06

TIME\_ELAPSED: 98

TIME\_RESET: NULL

STATUS: disabled

TYPE: status\_counter

COMMENT: Number of rows inserted

**Note**

Wildcard match is supported for counter and module names. For example, instead of specifying the full **dml\_inserts** counter name, you can specify **dml\_i%**. You can also enable, disable, or reset multiple counters or modules at once using a wildcard match. For example, specify **dml\_%** to enable, disable, or reset all counters that begin with **dml\_**.

After the counter is disabled, you can reset all counter values using the [**innodb\_monitor\_reset\_all**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_reset_all) option. All values are set to zero or NULL.

mysql> **SET GLOBAL innodb\_monitor\_reset\_all = dml\_inserts;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_METRICS WHERE NAME="dml\_inserts"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: dml\_inserts

SUBSYSTEM: dml

COUNT: 0

MAX\_COUNT: NULL

MIN\_COUNT: NULL

AVG\_COUNT: NULL

COUNT\_RESET: 0

MAX\_COUNT\_RESET: NULL

MIN\_COUNT\_RESET: NULL

AVG\_COUNT\_RESET: NULL

TIME\_ENABLED: NULL

TIME\_DISABLED: NULL

TIME\_ELAPSED: NULL

TIME\_RESET: NULL

STATUS: disabled

TYPE: status\_counter

COMMENT: Number of rows inserted

### 15.15.7 InnoDB INFORMATION\_SCHEMA Temporary Table Info Table

[**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) provides information about user-created **InnoDB** temporary tables that are active in the **InnoDB** instance. It does not provide information about internal **InnoDB** temporary tables used by the optimizer.

mysql> **SHOW TABLES FROM INFORMATION\_SCHEMA LIKE 'INNODB\_TEMP%';**

+---------------------------------------------+

| Tables\_in\_INFORMATION\_SCHEMA (INNODB\_TEMP%) |

+---------------------------------------------+

| INNODB\_TEMP\_TABLE\_INFO |

+---------------------------------------------+

For the table definition, see [Section 26.4.29, “The INFORMATION\_SCHEMA INNODB\_TEMP\_TABLE\_INFO Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table).

**Example 15.12 INNODB\_TEMP\_TABLE\_INFO**

This example demonstrates characteristics of the [**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) table.

Create a simple **InnoDB** temporary table:

mysql> **CREATE TEMPORARY TABLE t1 (c1 INT PRIMARY KEY) ENGINE=INNODB;**

Query [**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) to view the temporary table metadata.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TEMP\_TABLE\_INFO\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 194

NAME: #sql7a79\_1\_0

N\_COLS: 4

SPACE: 182

The **TABLE\_ID**is a unique identifier for the temporary table. The **NAME** column displays the system-generated name for the temporary table, which is prefixed with “#sql”. The number of columns (**N\_COLS**) is 4 rather than 1 because **InnoDB** always creates three hidden table columns (**DB\_ROW\_ID**, **DB\_TRX\_ID**, and **DB\_ROLL\_PTR**).

Restart MySQL and query [**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table).

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TEMP\_TABLE\_INFO\G**

An empty set is returned because [**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) and its data are not persisted to disk when the server is shut down.

Create a new temporary table.

mysql> **CREATE TEMPORARY TABLE t1 (c1 INT PRIMARY KEY) ENGINE=INNODB;**

Query [**INNODB\_TEMP\_TABLE\_INFO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-temp-table-info-table) to view the temporary table metadata.

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_TEMP\_TABLE\_INFO\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_ID: 196

NAME: #sql7b0e\_1\_0

N\_COLS: 4

SPACE: 184

The **SPACE** ID may be different because it is dynamically generated when the server is started.

### 15.15.8 Retrieving InnoDB Tablespace Metadata from INFORMATION\_SCHEMA.FILES

The [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table provides metadata about all **InnoDB** tablespace types including [file-per-table tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table), [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace), the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace), [temporary table tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_temporary_tablespace), and [undo tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_tablespace) (if present).

This section provides **InnoDB**-specific usage examples. For more information about data provided by the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table, see [Section 26.3.15, “The INFORMATION\_SCHEMA FILES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table).

**Note**

The [**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) and [**INNODB\_DATAFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-datafiles-table) tables also provide metadata about **InnoDB** tablespaces, but data is limited to file-per-table, general, and undo tablespaces.

This query retrieves metadata about the **InnoDB** system tablespace from fields of the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table that are pertinent to **InnoDB** tablespaces. [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) fields that are not relevant to **InnoDB** always return NULL, and are excluded from the query.

mysql> **SELECT FILE\_ID, FILE\_NAME, FILE\_TYPE, TABLESPACE\_NAME, FREE\_EXTENTS,**

**TOTAL\_EXTENTS, EXTENT\_SIZE, INITIAL\_SIZE, MAXIMUM\_SIZE, AUTOEXTEND\_SIZE, DATA\_FREE, STATUS ENGINE**

**FROM INFORMATION\_SCHEMA.FILES WHERE TABLESPACE\_NAME LIKE 'innodb\_system' \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_ID: 0

FILE\_NAME: ./ibdata1

FILE\_TYPE: TABLESPACE

TABLESPACE\_NAME: innodb\_system

FREE\_EXTENTS: 0

TOTAL\_EXTENTS: 12

EXTENT\_SIZE: 1048576

INITIAL\_SIZE: 12582912

MAXIMUM\_SIZE: NULL

AUTOEXTEND\_SIZE: 67108864

DATA\_FREE: 4194304

ENGINE: NORMAL

This query retrieves the **FILE\_ID** (equivalent to the space ID) and the **FILE\_NAME** (which includes path information) for **InnoDB** file-per-table and general tablespaces. File-per-table and general tablespaces have a .ibd file extension.

mysql> **SELECT FILE\_ID, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES**

**WHERE FILE\_NAME LIKE '%.ibd%' ORDER BY FILE\_ID;**

+---------+---------------------------------------+

| FILE\_ID | FILE\_NAME |

+---------+---------------------------------------+

| 2 | ./mysql/plugin.ibd |

| 3 | ./mysql/servers.ibd |

| 4 | ./mysql/help\_topic.ibd |

| 5 | ./mysql/help\_category.ibd |

| 6 | ./mysql/help\_relation.ibd |

| 7 | ./mysql/help\_keyword.ibd |

| 8 | ./mysql/time\_zone\_name.ibd |

| 9 | ./mysql/time\_zone.ibd |

| 10 | ./mysql/time\_zone\_transition.ibd |

| 11 | ./mysql/time\_zone\_transition\_type.ibd |

| 12 | ./mysql/time\_zone\_leap\_second.ibd |

| 13 | ./mysql/innodb\_table\_stats.ibd |

| 14 | ./mysql/innodb\_index\_stats.ibd |

| 15 | ./mysql/slave\_relay\_log\_info.ibd |

| 16 | ./mysql/slave\_master\_info.ibd |

| 17 | ./mysql/slave\_worker\_info.ibd |

| 18 | ./mysql/gtid\_executed.ibd |

| 19 | ./mysql/server\_cost.ibd |

| 20 | ./mysql/engine\_cost.ibd |

| 21 | ./sys/sys\_config.ibd |

| 23 | ./test/t1.ibd |

| 26 | /home/user/test/test/t2.ibd |

+---------+---------------------------------------+

This query retrieves the **FILE\_ID** and **FILE\_NAME** for the **InnoDB** global temporary tablespace. Global temporary tablespace file names are prefixed by **ibtmp**.

mysql> **SELECT FILE\_ID, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES**

**WHERE FILE\_NAME LIKE '%ibtmp%';**

+---------+-----------+

| FILE\_ID | FILE\_NAME |

+---------+-----------+

| 22 | ./ibtmp1 |

+---------+-----------+

Similarly, **InnoDB** undo tablespace file names are prefixed by **undo**. The following query returns the **FILE\_ID** and **FILE\_NAME** for **InnoDB** undo tablespaces.

mysql> **SELECT FILE\_ID, FILE\_NAME FROM INFORMATION\_SCHEMA.FILES**

**WHERE FILE\_NAME LIKE '%undo%';**

## 15.16 InnoDB Integration with MySQL Performance Schema

[15.16.1 Monitoring ALTER TABLE Progress for InnoDB Tables Using Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-alter-table-performance-schema)

[15.16.2 Monitoring InnoDB Mutex Waits Using Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-innodb-mutex-waits-performance-schema)

This section provides a brief introduction to **InnoDB** integration with Performance Schema. For comprehensive Performance Schema documentation, see [Chapter 27, *MySQL Performance Schema*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html).

You can profile certain internal **InnoDB** operations using the MySQL [Performance Schema feature](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html). This type of tuning is primarily for expert users who evaluate optimization strategies to overcome performance bottlenecks. DBAs can also use this feature for capacity planning, to see whether their typical workload encounters any performance bottlenecks with a particular combination of CPU, RAM, and disk storage; and if so, to judge whether performance can be improved by increasing the capacity of some part of the system.

To use this feature to examine **InnoDB** performance:

You must be generally familiar with how to use the [Performance Schema feature](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html). For example, you should know how enable instruments and consumers, and how to query **performance\_schema** tables to retrieve data. For an introductory overview, see [Section 27.1, “Performance Schema Quick Start”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-quick-start).

You should be familiar with Performance Schema instruments that are available for **InnoDB**. To view **InnoDB**-related instruments, you can query the [**setup\_instruments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-setup-instruments-table) table for instrument names that contain '**innodb**'.

mysql> **SELECT \***

**FROM performance\_schema.setup\_instruments**

**WHERE NAME LIKE '%innodb%';**

+-------------------------------------------------------+---------+-------+

| NAME | ENABLED | TIMED |

+-------------------------------------------------------+---------+-------+

| wait/synch/mutex/innodb/commit\_cond\_mutex | NO | NO |

| wait/synch/mutex/innodb/innobase\_share\_mutex | NO | NO |

| wait/synch/mutex/innodb/autoinc\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_zip\_mutex | NO | NO |

| wait/synch/mutex/innodb/cache\_last\_read\_mutex | NO | NO |

| wait/synch/mutex/innodb/dict\_foreign\_err\_mutex | NO | NO |

| wait/synch/mutex/innodb/dict\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/recalc\_pool\_mutex | NO | NO |

...

| wait/io/file/innodb/innodb\_data\_file | YES | YES |

| wait/io/file/innodb/innodb\_log\_file | YES | YES |

| wait/io/file/innodb/innodb\_temp\_file | YES | YES |

| stage/innodb/alter table (end) | YES | YES |

| stage/innodb/alter table (flush) | YES | YES |

| stage/innodb/alter table (insert) | YES | YES |

| stage/innodb/alter table (log apply index) | YES | YES |

| stage/innodb/alter table (log apply table) | YES | YES |

| stage/innodb/alter table (merge sort) | YES | YES |

| stage/innodb/alter table (read PK and internal sort) | YES | YES |

| stage/innodb/buffer pool load | YES | YES |

| memory/innodb/buf\_buf\_pool | NO | NO |

| memory/innodb/dict\_stats\_bg\_recalc\_pool\_t | NO | NO |

| memory/innodb/dict\_stats\_index\_map\_t | NO | NO |

| memory/innodb/dict\_stats\_n\_diff\_on\_level | NO | NO |

| memory/innodb/other | NO | NO |

| memory/innodb/row\_log\_buf | NO | NO |

| memory/innodb/row\_merge\_sort | NO | NO |

| memory/innodb/std | NO | NO |

| memory/innodb/sync\_debug\_latches | NO | NO |

| memory/innodb/trx\_sys\_t::rw\_trx\_ids | NO | NO |

...

+-------------------------------------------------------+---------+-------+

155 rows in set (0.00 sec)

For additional information about the instrumented **InnoDB** objects, you can query Performance Schema [instances tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-instance-tables), which provide additional information about instrumented objects. Instance tables relevant to **InnoDB** include:

The [**mutex\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-mutex-instances-table) table

The [**rwlock\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-rwlock-instances-table) table

The [**cond\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-cond-instances-table) table

The [**file\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-file-instances-table) table

**Note**

Mutexes and RW-locks related to the **InnoDB** buffer pool are not included in this coverage; the same applies to the output of the **SHOW ENGINE INNODB MUTEX** command.

For example, to view information about instrumented **InnoDB** file objects seen by the Performance Schema when executing file I/O instrumentation, you might issue the following query:

mysql> **SELECT \***

**FROM performance\_schema.file\_instances**

**WHERE EVENT\_NAME LIKE '%innodb%'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_NAME: /path/to/mysql-8.0/data/ibdata1

EVENT\_NAME: wait/io/file/innodb/innodb\_data\_file

OPEN\_COUNT: 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_NAME: /path/to/mysql-8.0/data/ib\_logfile0

EVENT\_NAME: wait/io/file/innodb/innodb\_log\_file

OPEN\_COUNT: 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_NAME: /path/to/mysql-8.0/data/ib\_logfile1

EVENT\_NAME: wait/io/file/innodb/innodb\_log\_file

OPEN\_COUNT: 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FILE\_NAME: /path/to/mysql-8.0/data/mysql/engine\_cost.ibd

EVENT\_NAME: wait/io/file/innodb/innodb\_data\_file

OPEN\_COUNT: 3

...

You should be familiar with **performance\_schema** tables that store **InnoDB** event data. Tables relevant to **InnoDB**-related events include:

The [Wait Event](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-wait-tables) tables, which store wait events.

The [Summary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-summary-tables) tables, which provide aggregated information for terminated events over time. Summary tables include [file I/O summary tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-file-summary-tables), which aggregate information about I/O operations.

[Stage Event](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-tables) tables, which store event data for **InnoDB** [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and buffer pool load operations. For more information, see [Section 15.16.1, “Monitoring ALTER TABLE Progress for InnoDB Tables Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-alter-table-performance-schema), and [Monitoring Buffer Pool Load Progress Using Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-buffer-pool-load-performance-schema).

If you are only interested in **InnoDB**-related objects, use the clause **WHERE EVENT\_NAME LIKE '%innodb%'** or **WHERE NAME LIKE '%innodb%'** (as required) when querying these tables.

### 15.16.1 Monitoring ALTER TABLE Progress for InnoDB Tables Using Performance Schema

You can monitor [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) progress for **InnoDB** tables using [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html).

There are seven stage events that represent different phases of [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Each stage event reports a running total of **WORK\_COMPLETED** and **WORK\_ESTIMATED** for the overall [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation as it progresses through its different phases. **WORK\_ESTIMATED** is calculated using a formula that takes into account all of the work that [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) performs, and may be revised during [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) processing. **WORK\_COMPLETED** and **WORK\_ESTIMATED** values are an abstract representation of all of the work performed by [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

In order of occurrence, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) stage events include:

**stage/innodb/alter table (read PK and internal sort)**: This stage is active when [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is in the reading-primary-key phase. It starts with **WORK\_COMPLETED=0** and **WORK\_ESTIMATED** set to the estimated number of pages in the primary key. When the stage is completed, **WORK\_ESTIMATED** is updated to the actual number of pages in the primary key.

**stage/innodb/alter table (merge sort)**: This stage is repeated for each index added by the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation.

**stage/innodb/alter table (insert)**: This stage is repeated for each index added by the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation.

**stage/innodb/alter table (log apply index)**: This stage includes the application of DML log generated while [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) was running.

**stage/innodb/alter table (flush)**: Before this stage begins, **WORK\_ESTIMATED** is updated with a more accurate estimate, based on the length of the flush list.

**stage/innodb/alter table (log apply table)**: This stage includes the application of concurrent DML log generated while [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) was running. The duration of this phase depends on the extent of table changes. This phase is instant if no concurrent DML was run on the table.

**stage/innodb/alter table (end)**: Includes any remaining work that appeared after the flush phase, such as reapplying DML that was executed on the table while [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) was running.

**Note**

**InnoDB** [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) stage events do not currently account for the addition of spatial indexes.

#### ALTER TABLE Monitoring Example Using Performance Schema

The following example demonstrates how to enable the **stage/innodb/alter table%** stage event instruments and related consumer tables to monitor [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) progress. For information about Performance Schema stage event instruments and related consumers, see [Section 27.12.5, “Performance Schema Stage Event Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-tables).

Enable the **stage/innodb/alter%** instruments:

mysql> **UPDATE performance\_schema.setup\_instruments**

**SET ENABLED = 'YES'**

**WHERE NAME LIKE 'stage/innodb/alter%';**

Query OK, 7 rows affected (0.00 sec)

Rows matched: 7 Changed: 7 Warnings: 0

Enable the stage event consumer tables, which include [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table), [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table), and [**events\_stages\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-long-table).

mysql> **UPDATE performance\_schema.setup\_consumers**

**SET ENABLED = 'YES'**

**WHERE NAME LIKE '%stages%';**

Query OK, 3 rows affected (0.00 sec)

Rows matched: 3 Changed: 3 Warnings: 0

Run an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation. In this example, a **middle\_name** column is added to the employees table of the employees sample database.

mysql> **ALTER TABLE employees.employees ADD COLUMN middle\_name varchar(14) AFTER first\_name;**

Query OK, 0 rows affected (9.27 sec)

Records: 0 Duplicates: 0 Warnings: 0

Check the progress of the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation by querying the Performance Schema [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table. The stage event shown differs depending on which [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) phase is currently in progress. The **WORK\_COMPLETED** column shows the work completed. The **WORK\_ESTIMATED** column provides an estimate of the remaining work.

mysql> **SELECT EVENT\_NAME, WORK\_COMPLETED, WORK\_ESTIMATED**

**FROM performance\_schema.events\_stages\_current;**

+------------------------------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+------------------------------------------------------+----------------+----------------+

| stage/innodb/alter table (read PK and internal sort) | 280 | 1245 |

+------------------------------------------------------+----------------+----------------+

1 row in set (0.01 sec)

The [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table returns an empty set if the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation has completed. In this case, you can check the [**events\_stages\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-history-table) table to view event data for the completed operation. For example:

mysql> **SELECT EVENT\_NAME, WORK\_COMPLETED, WORK\_ESTIMATED**

**FROM performance\_schema.events\_stages\_history;**

+------------------------------------------------------+----------------+----------------+

| EVENT\_NAME | WORK\_COMPLETED | WORK\_ESTIMATED |

+------------------------------------------------------+----------------+----------------+

| stage/innodb/alter table (read PK and internal sort) | 886 | 1213 |

| stage/innodb/alter table (flush) | 1213 | 1213 |

| stage/innodb/alter table (log apply table) | 1597 | 1597 |

| stage/innodb/alter table (end) | 1597 | 1597 |

| stage/innodb/alter table (log apply table) | 1981 | 1981 |

+------------------------------------------------------+----------------+----------------+

5 rows in set (0.00 sec)

As shown above, the **WORK\_ESTIMATED** value was revised during **ALTER TABLE** processing. The estimated work after completion of the initial stage is 1213. When **ALTER TABLE** processing completed, **WORK\_ESTIMATED** was set to the actual value, which is 1981.

### 15.16.2 Monitoring InnoDB Mutex Waits Using Performance Schema

A mutex is a synchronization mechanism used in the code to enforce that only one thread at a given time can have access to a common resource. When two or more threads executing in the server need to access the same resource, the threads compete against each other. The first thread to obtain a lock on the mutex causes the other threads to wait until the lock is released.

For **InnoDB** mutexes that are instrumented, mutex waits can be monitored using [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html). Wait event data collected in Performance Schema tables can help identify mutexes with the most waits or the greatest total wait time, for example.

The following example demonstrates how to enable **InnoDB** mutex wait instruments, how to enable associated consumers, and how to query wait event data.

To view available **InnoDB** mutex wait instruments, query the Performance Schema [**setup\_instruments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-setup-instruments-table) table. All **InnoDB** mutex wait instruments are disabled by default.

mysql> **SELECT \***

**FROM performance\_schema.setup\_instruments**

**WHERE NAME LIKE '%wait/synch/mutex/innodb%';**

+---------------------------------------------------------+---------+-------+

| NAME | ENABLED | TIMED |

+---------------------------------------------------------+---------+-------+

| wait/synch/mutex/innodb/commit\_cond\_mutex | NO | NO |

| wait/synch/mutex/innodb/innobase\_share\_mutex | NO | NO |

| wait/synch/mutex/innodb/autoinc\_mutex | NO | NO |

| wait/synch/mutex/innodb/autoinc\_persisted\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_flush\_state\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_LRU\_list\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_free\_list\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_zip\_free\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_zip\_hash\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_pool\_zip\_mutex | NO | NO |

| wait/synch/mutex/innodb/cache\_last\_read\_mutex | NO | NO |

| wait/synch/mutex/innodb/dict\_foreign\_err\_mutex | NO | NO |

| wait/synch/mutex/innodb/dict\_persist\_dirty\_tables\_mutex | NO | NO |

| wait/synch/mutex/innodb/dict\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/recalc\_pool\_mutex | NO | NO |

| wait/synch/mutex/innodb/fil\_system\_mutex | NO | NO |

| wait/synch/mutex/innodb/flush\_list\_mutex | NO | NO |

| wait/synch/mutex/innodb/fts\_bg\_threads\_mutex | NO | NO |

| wait/synch/mutex/innodb/fts\_delete\_mutex | NO | NO |

| wait/synch/mutex/innodb/fts\_optimize\_mutex | NO | NO |

| wait/synch/mutex/innodb/fts\_doc\_id\_mutex | NO | NO |

| wait/synch/mutex/innodb/log\_flush\_order\_mutex | NO | NO |

| wait/synch/mutex/innodb/hash\_table\_mutex | NO | NO |

| wait/synch/mutex/innodb/ibuf\_bitmap\_mutex | NO | NO |

| wait/synch/mutex/innodb/ibuf\_mutex | NO | NO |

| wait/synch/mutex/innodb/ibuf\_pessimistic\_insert\_mutex | NO | NO |

| wait/synch/mutex/innodb/log\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/log\_sys\_write\_mutex | NO | NO |

| wait/synch/mutex/innodb/mutex\_list\_mutex | NO | NO |

| wait/synch/mutex/innodb/page\_zip\_stat\_per\_index\_mutex | NO | NO |

| wait/synch/mutex/innodb/purge\_sys\_pq\_mutex | NO | NO |

| wait/synch/mutex/innodb/recv\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/recv\_writer\_mutex | NO | NO |

| wait/synch/mutex/innodb/redo\_rseg\_mutex | NO | NO |

| wait/synch/mutex/innodb/noredo\_rseg\_mutex | NO | NO |

| wait/synch/mutex/innodb/rw\_lock\_list\_mutex | NO | NO |

| wait/synch/mutex/innodb/rw\_lock\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_dict\_tmpfile\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_innodb\_monitor\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_misc\_tmpfile\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_monitor\_file\_mutex | NO | NO |

| wait/synch/mutex/innodb/buf\_dblwr\_mutex | NO | NO |

| wait/synch/mutex/innodb/trx\_undo\_mutex | NO | NO |

| wait/synch/mutex/innodb/trx\_pool\_mutex | NO | NO |

| wait/synch/mutex/innodb/trx\_pool\_manager\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/lock\_mutex | NO | NO |

| wait/synch/mutex/innodb/lock\_wait\_mutex | NO | NO |

| wait/synch/mutex/innodb/trx\_mutex | NO | NO |

| wait/synch/mutex/innodb/srv\_threads\_mutex | NO | NO |

| wait/synch/mutex/innodb/rtr\_active\_mutex | NO | NO |

| wait/synch/mutex/innodb/rtr\_match\_mutex | NO | NO |

| wait/synch/mutex/innodb/rtr\_path\_mutex | NO | NO |

| wait/synch/mutex/innodb/rtr\_ssn\_mutex | NO | NO |

| wait/synch/mutex/innodb/trx\_sys\_mutex | NO | NO |

| wait/synch/mutex/innodb/zip\_pad\_mutex | NO | NO |

| wait/synch/mutex/innodb/master\_key\_id\_mutex | NO | NO |

+---------------------------------------------------------+---------+-------+

Some **InnoDB** mutex instances are created at server startup and are only instrumented if the associated instrument is also enabled at server startup. To ensure that all **InnoDB** mutex instances are instrumented and enabled, add the following **performance-schema-instrument** rule to your MySQL configuration file:

performance-schema-instrument='wait/synch/mutex/innodb/%=ON'

If you do not require wait event data for all **InnoDB** mutexes, you can disable specific instruments by adding additional **performance-schema-instrument** rules to your MySQL configuration file. For example, to disable **InnoDB** mutex wait event instruments related to full-text search, add the following rule:

performance-schema-instrument='wait/synch/mutex/innodb/fts%=OFF'

**Note**

Rules with a longer prefix such as **wait/synch/mutex/innodb/fts%** take precedence over rules with shorter prefixes such as **wait/synch/mutex/innodb/%**.

After adding the **performance-schema-instrument** rules to your configuration file, restart the server. All the **InnoDB** mutexes except for those related to full text search are enabled. To verify, query the [**setup\_instruments**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-setup-instruments-table) table. The **ENABLED** and **TIMED** columns should be set to **YES** for the instruments that you enabled.

mysql> **SELECT \***

**FROM performance\_schema.setup\_instruments**

**WHERE NAME LIKE '%wait/synch/mutex/innodb%';**

+-------------------------------------------------------+---------+-------+

| NAME | ENABLED | TIMED |

+-------------------------------------------------------+---------+-------+

| wait/synch/mutex/innodb/commit\_cond\_mutex | YES | YES |

| wait/synch/mutex/innodb/innobase\_share\_mutex | YES | YES |

| wait/synch/mutex/innodb/autoinc\_mutex | YES | YES |

...

| wait/synch/mutex/innodb/master\_key\_id\_mutex | YES | YES |

+-------------------------------------------------------+---------+-------+

49 rows in set (0.00 sec)

Enable wait event consumers by updating the [**setup\_consumers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-setup-consumers-table) table. Wait event consumers are disabled by default.

mysql> **UPDATE performance\_schema.setup\_consumers**

**SET enabled = 'YES'**

**WHERE name like 'events\_waits%';**

Query OK, 3 rows affected (0.00 sec)

Rows matched: 3 Changed: 3 Warnings: 0

You can verify that wait event consumers are enabled by querying the [**setup\_consumers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-setup-consumers-table) table. The [**events\_waits\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-current-table), [**events\_waits\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-history-table), and [**events\_waits\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-history-long-table) consumers should be enabled.

mysql> **SELECT \* FROM performance\_schema.setup\_consumers;**

+----------------------------------+---------+

| NAME | ENABLED |

+----------------------------------+---------+

| events\_stages\_current | NO |

| events\_stages\_history | NO |

| events\_stages\_history\_long | NO |

| events\_statements\_current | YES |

| events\_statements\_history | YES |

| events\_statements\_history\_long | NO |

| events\_transactions\_current | YES |

| events\_transactions\_history | YES |

| events\_transactions\_history\_long | NO |

| events\_waits\_current | YES |

| events\_waits\_history | YES |

| events\_waits\_history\_long | YES |

| global\_instrumentation | YES |

| thread\_instrumentation | YES |

| statements\_digest | YES |

+----------------------------------+---------+

15 rows in set (0.00 sec)

Once instruments and consumers are enabled, run the workload that you want to monitor. In this example, the [**mysqlslap**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlslap) load emulation client is used to simulate a workload.

shell> **./mysqlslap --auto-generate-sql --concurrency=100 --iterations=10**

**--number-of-queries=1000 --number-char-cols=6 --number-int-cols=6;**

Query the wait event data. In this example, wait event data is queried from the [**events\_waits\_summary\_global\_by\_event\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-wait-summary-tables) table which aggregates data found in the [**events\_waits\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-current-table), [**events\_waits\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-history-table), and [**events\_waits\_history\_long**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-waits-history-long-table) tables. Data is summarized by event name (**EVENT\_NAME**), which is the name of the instrument that produced the event. Summarized data includes:

**COUNT\_STAR**

The number of summarized wait events.

**SUM\_TIMER\_WAIT**

The total wait time of the summarized timed wait events.

**MIN\_TIMER\_WAIT**

The minimum wait time of the summarized timed wait events.

**AVG\_TIMER\_WAIT**

The average wait time of the summarized timed wait events.

**MAX\_TIMER\_WAIT**

The maximum wait time of the summarized timed wait events.

The following query returns the instrument name (**EVENT\_NAME**), the number of wait events (**COUNT\_STAR**), and the total wait time for the events for that instrument (**SUM\_TIMER\_WAIT**). Because waits are timed in picoseconds (trillionths of a second) by default, wait times are divided by 1000000000 to show wait times in milliseconds. Data is presented in descending order, by the number of summarized wait events (**COUNT\_STAR**). You can adjust the **ORDER BY** clause to order the data by total wait time.

mysql> **SELECT EVENT\_NAME, COUNT\_STAR, SUM\_TIMER\_WAIT/1000000000 SUM\_TIMER\_WAIT\_MS**

**FROM performance\_schema.events\_waits\_summary\_global\_by\_event\_name**

**WHERE SUM\_TIMER\_WAIT > 0 AND EVENT\_NAME LIKE 'wait/synch/mutex/innodb/%'**

**ORDER BY COUNT\_STAR DESC;**

+---------------------------------------------------------+------------+-------------------+

| EVENT\_NAME | COUNT\_STAR | SUM\_TIMER\_WAIT\_MS |

+---------------------------------------------------------+------------+-------------------+

| wait/synch/mutex/innodb/trx\_mutex | 201111 | 23.4719 |

| wait/synch/mutex/innodb/fil\_system\_mutex | 62244 | 9.6426 |

| wait/synch/mutex/innodb/redo\_rseg\_mutex | 48238 | 3.1135 |

| wait/synch/mutex/innodb/log\_sys\_mutex | 46113 | 2.0434 |

| wait/synch/mutex/innodb/trx\_sys\_mutex | 35134 | 1068.1588 |

| wait/synch/mutex/innodb/lock\_mutex | 34872 | 1039.2589 |

| wait/synch/mutex/innodb/log\_sys\_write\_mutex | 17805 | 1526.0490 |

| wait/synch/mutex/innodb/dict\_sys\_mutex | 14912 | 1606.7348 |

| wait/synch/mutex/innodb/trx\_undo\_mutex | 10634 | 1.1424 |

| wait/synch/mutex/innodb/rw\_lock\_list\_mutex | 8538 | 0.1960 |

| wait/synch/mutex/innodb/buf\_pool\_free\_list\_mutex | 5961 | 0.6473 |

| wait/synch/mutex/innodb/trx\_pool\_mutex | 4885 | 8821.7496 |

| wait/synch/mutex/innodb/buf\_pool\_LRU\_list\_mutex | 4364 | 0.2077 |

| wait/synch/mutex/innodb/innobase\_share\_mutex | 3212 | 0.2650 |

| wait/synch/mutex/innodb/flush\_list\_mutex | 3178 | 0.2349 |

| wait/synch/mutex/innodb/trx\_pool\_manager\_mutex | 2495 | 0.1310 |

| wait/synch/mutex/innodb/buf\_pool\_flush\_state\_mutex | 1318 | 0.2161 |

| wait/synch/mutex/innodb/log\_flush\_order\_mutex | 1250 | 0.0893 |

| wait/synch/mutex/innodb/buf\_dblwr\_mutex | 951 | 0.0918 |

| wait/synch/mutex/innodb/recalc\_pool\_mutex | 670 | 0.0942 |

| wait/synch/mutex/innodb/dict\_persist\_dirty\_tables\_mutex | 345 | 0.0414 |

| wait/synch/mutex/innodb/lock\_wait\_mutex | 303 | 0.1565 |

| wait/synch/mutex/innodb/autoinc\_mutex | 196 | 0.0213 |

| wait/synch/mutex/innodb/autoinc\_persisted\_mutex | 196 | 0.0175 |

| wait/synch/mutex/innodb/purge\_sys\_pq\_mutex | 117 | 0.0308 |

| wait/synch/mutex/innodb/srv\_sys\_mutex | 94 | 0.0077 |

| wait/synch/mutex/innodb/ibuf\_mutex | 22 | 0.0086 |

| wait/synch/mutex/innodb/recv\_sys\_mutex | 12 | 0.0008 |

| wait/synch/mutex/innodb/srv\_innodb\_monitor\_mutex | 4 | 0.0009 |

| wait/synch/mutex/innodb/recv\_writer\_mutex | 1 | 0.0005 |

+---------------------------------------------------------+------------+-------------------+

**Note**

The preceding result set includes wait event data produced during the startup process. To exclude this data, you can truncate the [**events\_waits\_summary\_global\_by\_event\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-wait-summary-tables) table immediately after startup and before running your workload. However, the truncate operation itself may produce a negligible amount wait event data.

mysql> **TRUNCATE performance\_schema.events\_waits\_summary\_global\_by\_event\_name;**

## 15.17 InnoDB Monitors

[15.17.1 InnoDB Monitor Types](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-monitor-types)

[15.17.2 Enabling InnoDB Monitors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-enabling-monitors)

[15.17.3 InnoDB Standard Monitor and Lock Monitor Output](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-standard-monitor)

**InnoDB** monitors provide information about the **InnoDB** internal state. This information is useful for performance tuning.

### 15.17.1 InnoDB Monitor Types

There are two types of **InnoDB** monitor:

The standard **InnoDB** Monitor displays the following types of information:

Work done by the main background thread

Semaphore waits

Data about the most recent foreign key and deadlock errors

Lock waits for transactions

Table and record locks held by active transactions

Pending I/O operations and related statistics

Insert buffer and adaptive hash index statistics

Redo log data

Buffer pool statistics

Row operation data

The **InnoDB** Lock Monitor prints additional lock information as part of the standard **InnoDB** Monitor output.

### 15.17.2 Enabling InnoDB Monitors

When **InnoDB** monitors are enabled for periodic output, **InnoDB** writes the output to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) server standard error output (**stderr**) every 15 seconds, approximately.

**InnoDB** sends the monitor output to **stderr** rather than to **stdout** or fixed-size memory buffers to avoid potential buffer overflows.

On Windows, **stderr** is directed to the default log file unless configured otherwise. If you want to direct the output to the console window rather than to the error log, start the server from a command prompt in a console window with the [--console](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_console) option. For more information, see [Default Error Log Destination on Windows](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log-destination-configuration-windows).

On Unix and Unix-like systems, **stderr** is typically directed to the terminal unless configured otherwise. For more information, see [Default Error Log Destination on Unix and Unix-Like Systems](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log-destination-configuration-unix).

**InnoDB** monitors should only be enabled when you actually want to see monitor information because output generation causes some performance decrement. Also, if monitor output is directed to the error log, the log may become quite large if you forget to disable the monitor later.

**Note**

To assist with troubleshooting, **InnoDB** temporarily enables standard **InnoDB** Monitor output under certain conditions. For more information, see [Section 15.21, “InnoDB Troubleshooting”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-troubleshooting).

**InnoDB** monitor output begins with a header containing a timestamp and the monitor name. For example:

=====================================

2014-10-16 18:37:29 0x7fc2a95c1700 INNODB MONITOR OUTPUT

=====================================

The header for the standard **InnoDB** Monitor (**INNODB MONITOR OUTPUT**) is also used for the Lock Monitor because the latter produces the same output with the addition of extra lock information.

The [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) and [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) system variables are used to enable the standard **InnoDB** Monitor and **InnoDB** Lock Monitor.

The [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege is required to enable or disable **InnoDB** Monitors.

#### Enabling the Standard InnoDB Monitor

Enable the standard **InnoDB** Monitor by setting the [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) system variable to **ON**.

SET GLOBAL innodb\_status\_output=ON;

To disable the standard **InnoDB** Monitor, set [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) to **OFF**.

When you shut down the server, the [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) variable is set to the default **OFF** value.

#### Enabling the InnoDB Lock Monitor

**InnoDB** Lock Monitor data is printed with the **InnoDB** Standard Monitor output. Both the **InnoDB** Standard Monitor and **InnoDB** Lock Monitor must be enabled to have **InnoDB** Lock Monitor data printed periodically.

To enable the **InnoDB** Lock Monitor, set the [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) system variable to **ON**. Both the **InnoDB** standard Monitor and **InnoDB** Lock Monitor must be enabled to have **InnoDB** Lock Monitor data printed periodically:

SET GLOBAL innodb\_status\_output=ON;

SET GLOBAL innodb\_status\_output\_locks=ON;

To disable the **InnoDB** Lock Monitor, set [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) to **OFF**. Set [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) to **OFF** to also disable the **InnoDB** Standard Monitor.

When you shut down the server, the [**innodb\_status\_output**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output) and [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks) variables are set to the default **OFF** value.

**Note**

To enable the **InnoDB** Lock Monitor for [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output, you are only required to enable [**innodb\_status\_output\_locks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_status_output_locks).

#### Obtaining Standard InnoDB Monitor Output On Demand

As an alternative to enabling the standard **InnoDB** Monitor for periodic output, you can obtain standard **InnoDB** Monitor output on demand using the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) SQL statement, which fetches the output to your client program. If you are using the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) interactive client, the output is more readable if you replace the usual semicolon statement terminator with **\G**:

mysql> **SHOW ENGINE INNODB STATUS\G**

[**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output also includes **InnoDB** Lock Monitor data if the **InnoDB** Lock Monitor is enabled.

#### Directing Standard InnoDB Monitor Output to a Status File

Standard **InnoDB** Monitor output can be enabled and directed to a status file by specifying the --innodb-status-file option at startup. When this option is used, **InnoDB** creates a file named innodb\_status.***pid*** in the data directory and writes output to it every 15 seconds, approximately.

**InnoDB** removes the status file when the server is shut down normally. If an abnormal shutdown occurs, the status file may have to be removed manually.

The --innodb-status-file option is intended for temporary use, as output generation can affect performance, and the innodb\_status.***pid*** file can become quite large over time.

### 15.17.3 InnoDB Standard Monitor and Lock Monitor Output

The Lock Monitor is the same as the Standard Monitor except that it includes additional lock information. Enabling either monitor for periodic output turns on the same output stream, but the stream includes extra information if the Lock Monitor is enabled. For example, if you enable the Standard Monitor and Lock Monitor, that turns on a single output stream. The stream includes extra lock information until you disable the Lock Monitor.

Standard Monitor output is limited to 1MB when produced using the [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) statement. This limit does not apply to output written to server standard error output (**stderr**).

Example Standard Monitor output:

mysql> **SHOW ENGINE INNODB STATUS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Type: InnoDB

Name:

Status:

=====================================

2018-04-12 15:14:08 0x7f971c063700 INNODB MONITOR OUTPUT

=====================================

Per second averages calculated from the last 4 seconds

-----------------

BACKGROUND THREAD

-----------------

srv\_master\_thread loops: 15 srv\_active, 0 srv\_shutdown, 1122 srv\_idle

srv\_master\_thread log flush and writes: 0

----------

SEMAPHORES

----------

OS WAIT ARRAY INFO: reservation count 24

OS WAIT ARRAY INFO: signal count 24

RW-shared spins 4, rounds 8, OS waits 4

RW-excl spins 2, rounds 60, OS waits 2

RW-sx spins 0, rounds 0, OS waits 0

Spin rounds per wait: 2.00 RW-shared, 30.00 RW-excl, 0.00 RW-sx

------------------------

LATEST FOREIGN KEY ERROR

------------------------

2018-04-12 14:57:24 0x7f97a9c91700 Transaction:

TRANSACTION 7717, ACTIVE 0 sec inserting

mysql tables in use 1, locked 1

4 lock struct(s), heap size 1136, 3 row lock(s), undo log entries 3

MySQL thread id 8, OS thread handle 140289365317376, query id 14 localhost root update

INSERT INTO child VALUES (NULL, 1), (NULL, 2), (NULL, 3), (NULL, 4), (NULL, 5), (NULL, 6)

Foreign key constraint fails for table `test`.`child`:

,

CONSTRAINT `child\_ibfk\_1` FOREIGN KEY (`parent\_id`) REFERENCES `parent` (`id`) ON DELETE

CASCADE ON UPDATE CASCADE

Trying to add in child table, in index par\_ind tuple:

DATA TUPLE: 2 fields;

0: len 4; hex 80000003; asc ;;

1: len 4; hex 80000003; asc ;;

But in parent table `test`.`parent`, in index PRIMARY,

the closest match we can find is record:

PHYSICAL RECORD: n\_fields 3; compact format; info bits 0

0: len 4; hex 80000004; asc ;;

1: len 6; hex 000000001e19; asc ;;

2: len 7; hex 81000001110137; asc 7;;

------------

TRANSACTIONS

------------

Trx id counter 7748

Purge done for trx's n:o < 7747 undo n:o < 0 state: running but idle

History list length 19

LIST OF TRANSACTIONS FOR EACH SESSION:

---TRANSACTION 421764459790000, not started

0 lock struct(s), heap size 1136, 0 row lock(s)

---TRANSACTION 7747, ACTIVE 23 sec starting index read

mysql tables in use 1, locked 1

LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)

MySQL thread id 9, OS thread handle 140286987249408, query id 51 localhost root updating

DELETE FROM t WHERE i = 1

------- TRX HAS BEEN WAITING 23 SEC FOR THIS LOCK TO BE GRANTED:

RECORD LOCKS space id 4 page no 4 n bits 72 index GEN\_CLUST\_INDEX of table `test`.`t`

trx id 7747 lock\_mode X waiting

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 6; hex 000000000202; asc ;;

1: len 6; hex 000000001e41; asc A;;

2: len 7; hex 820000008b0110; asc ;;

3: len 4; hex 80000001; asc ;;

------------------

TABLE LOCK table `test`.`t` trx id 7747 lock mode IX

RECORD LOCKS space id 4 page no 4 n bits 72 index GEN\_CLUST\_INDEX of table `test`.`t`

trx id 7747 lock\_mode X waiting

Record lock, heap no 3 PHYSICAL RECORD: n\_fields 4; compact format; info bits 0

0: len 6; hex 000000000202; asc ;;

1: len 6; hex 000000001e41; asc A;;

2: len 7; hex 820000008b0110; asc ;;

3: len 4; hex 80000001; asc ;;

--------

FILE I/O

--------

I/O thread 0 state: waiting for i/o request (insert buffer thread)

I/O thread 1 state: waiting for i/o request (log thread)

I/O thread 2 state: waiting for i/o request (read thread)

I/O thread 3 state: waiting for i/o request (read thread)

I/O thread 4 state: waiting for i/o request (read thread)

I/O thread 5 state: waiting for i/o request (read thread)

I/O thread 6 state: waiting for i/o request (write thread)

I/O thread 7 state: waiting for i/o request (write thread)

I/O thread 8 state: waiting for i/o request (write thread)

I/O thread 9 state: waiting for i/o request (write thread)

Pending normal aio reads: [0, 0, 0, 0] , aio writes: [0, 0, 0, 0] ,

ibuf aio reads:, log i/o's:, sync i/o's:

Pending flushes (fsync) log: 0; buffer pool: 0

833 OS file reads, 605 OS file writes, 208 OS fsyncs

0.00 reads/s, 0 avg bytes/read, 0.00 writes/s, 0.00 fsyncs/s

-------------------------------------

INSERT BUFFER AND ADAPTIVE HASH INDEX

-------------------------------------

Ibuf: size 1, free list len 0, seg size 2, 0 merges

merged operations:

insert 0, delete mark 0, delete 0

discarded operations:

insert 0, delete mark 0, delete 0

Hash table size 553253, node heap has 0 buffer(s)

Hash table size 553253, node heap has 1 buffer(s)

Hash table size 553253, node heap has 3 buffer(s)

Hash table size 553253, node heap has 0 buffer(s)

Hash table size 553253, node heap has 0 buffer(s)

Hash table size 553253, node heap has 0 buffer(s)

Hash table size 553253, node heap has 0 buffer(s)

Hash table size 553253, node heap has 0 buffer(s)

0.00 hash searches/s, 0.00 non-hash searches/s

---

LOG

---

Log sequence number 19643450

Log buffer assigned up to 19643450

Log buffer completed up to 19643450

Log written up to 19643450

Log flushed up to 19643450

Added dirty pages up to 19643450

Pages flushed up to 19643450

Last checkpoint at 19643450

129 log i/o's done, 0.00 log i/o's/second

----------------------

BUFFER POOL AND MEMORY

----------------------

Total large memory allocated 2198863872

Dictionary memory allocated 409606

Buffer pool size 131072

Free buffers 130095

Database pages 973

Old database pages 0

Modified db pages 0

Pending reads 0

Pending writes: LRU 0, flush list 0, single page 0

Pages made young 0, not young 0

0.00 youngs/s, 0.00 non-youngs/s

Pages read 810, created 163, written 404

0.00 reads/s, 0.00 creates/s, 0.00 writes/s

Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000

Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s

LRU len: 973, unzip\_LRU len: 0

I/O sum[0]:cur[0], unzip sum[0]:cur[0]

----------------------

INDIVIDUAL BUFFER POOL INFO

----------------------

---BUFFER POOL 0

Buffer pool size 65536

Free buffers 65043

Database pages 491

Old database pages 0

Modified db pages 0

Pending reads 0

Pending writes: LRU 0, flush list 0, single page 0

Pages made young 0, not young 0

0.00 youngs/s, 0.00 non-youngs/s

Pages read 411, created 80, written 210

0.00 reads/s, 0.00 creates/s, 0.00 writes/s

Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000

Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s

LRU len: 491, unzip\_LRU len: 0

I/O sum[0]:cur[0], unzip sum[0]:cur[0]

---BUFFER POOL 1

Buffer pool size 65536

Free buffers 65052

Database pages 482

Old database pages 0

Modified db pages 0

Pending reads 0

Pending writes: LRU 0, flush list 0, single page 0

Pages made young 0, not young 0

0.00 youngs/s, 0.00 non-youngs/s

Pages read 399, created 83, written 194

0.00 reads/s, 0.00 creates/s, 0.00 writes/s

No buffer pool page gets since the last printout

Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s

LRU len: 482, unzip\_LRU len: 0

I/O sum[0]:cur[0], unzip sum[0]:cur[0]

--------------

ROW OPERATIONS

--------------

0 queries inside InnoDB, 0 queries in queue

0 read views open inside InnoDB

Process ID=5772, Main thread ID=140286437054208 , state=sleeping

Number of rows inserted 57, updated 354, deleted 4, read 4421

0.00 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s

----------------------------

END OF INNODB MONITOR OUTPUT

============================

#### Standard Monitor Output Sections

For a description of each metric reported by the Standard Monitor, refer to the [Metrics](http://dev.mysql.com/doc/mysql-em-plugin/en/myoem-metrics.html) chapter in the [Oracle Enterprise Manager for MySQL Database User's Guide](http://dev.mysql.com/doc/mysql-em-plugin/en/).

**Status**

This section shows the timestamp, the monitor name, and the number of seconds that per-second averages are based on. The number of seconds is the elapsed time between the current time and the last time **InnoDB** Monitor output was printed.

**BACKGROUND THREAD**

The **srv\_master\_thread** lines shows work done by the main background thread.

**SEMAPHORES**

This section reports threads waiting for a semaphore and statistics on how many times threads have needed a spin or a wait on a mutex or a rw-lock semaphore. A large number of threads waiting for semaphores may be a result of disk I/O, or contention problems inside **InnoDB**. Contention can be due to heavy parallelism of queries or problems in operating system thread scheduling. Setting the [**innodb\_thread\_concurrency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_thread_concurrency) system variable smaller than the default value might help in such situations. The **Spin rounds per wait** line shows the number of spinlock rounds per OS wait for a mutex.

Mutex metrics are reported by [**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine).

**LATEST FOREIGN KEY ERROR**

This section provides information about the most recent foreign key constraint error. It is not present if no such error has occurred. The contents include the statement that failed as well as information about the constraint that failed and the referenced and referencing tables.

**LATEST DETECTED DEADLOCK**

This section provides information about the most recent deadlock. It is not present if no deadlock has occurred. The contents show which transactions are involved, the statement each was attempting to execute, the locks they have and need, and which transaction **InnoDB** decided to roll back to break the deadlock. The lock modes reported in this section are explained in [Section 15.7.1, “InnoDB Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking).

**TRANSACTIONS**

If this section reports lock waits, your applications might have lock contention. The output can also help to trace the reasons for transaction deadlocks.

**FILE I/O**

This section provides information about threads that **InnoDB** uses to perform various types of I/O. The first few of these are dedicated to general **InnoDB** processing. The contents also display information for pending I/O operations and statistics for I/O performance.

The number of these threads are controlled by the [**innodb\_read\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_io_threads) and [**innodb\_write\_io\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_write_io_threads) parameters. See [Section 15.14, “InnoDB Startup Options and System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-parameters).

**INSERT BUFFER AND ADAPTIVE HASH INDEX**

This section shows the status of the **InnoDB** insert buffer (also referred to as the [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer)) and the adaptive hash index.

For related information, see [Section 15.5.2, “Change Buffer”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-change-buffer), and [Section 15.5.3, “Adaptive Hash Index”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-adaptive-hash).

**LOG**

This section displays information about the **InnoDB** log. The contents include the current log sequence number, how far the log has been flushed to disk, and the position at which **InnoDB** last took a checkpoint. (See [Section 15.11.3, “InnoDB Checkpoints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-checkpoints).) The section also displays information about pending writes and write performance statistics.

**BUFFER POOL AND MEMORY**

This section gives you statistics on pages read and written. You can calculate from these numbers how many data file I/O operations your queries currently are doing.

For buffer pool statistics descriptions, see [Monitoring the Buffer Pool Using the InnoDB Standard Monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool-monitoring). For additional information about the operation of the buffer pool, see [Section 15.5.1, “Buffer Pool”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-buffer-pool).

**ROW OPERATIONS**

This section shows what the main thread is doing, including the number and performance rate for each type of row operation.

## 15.18 InnoDB Backup and Recovery

[15.18.1 InnoDB Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-backup)

[15.18.2 InnoDB Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery)

This section covers topics related to **InnoDB** backup and recovery.

For information about backup techniques applicable to **InnoDB**, see [Section 15.18.1, “InnoDB Backup”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-backup).

For information about point-in-time recovery, recovery from disk failure or corruption, and how **InnoDB** performs crash recovery, see [Section 15.18.2, “InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery).

### 15.18.1 InnoDB Backup

The key to safe database management is making regular backups. Depending on your data volume, number of MySQL servers, and database workload, you can use these backup techniques, alone or in combination: [hot backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_hot_backup) with MySQL Enterprise Backup; [cold backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cold_backup) by copying files while the MySQL server is shut down; [logical backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_logical_backup) with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) for smaller data volumes or to record the structure of schema objects. Hot and cold backups are [physical backups](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_physical_backup) that copy actual data files, which can be used directly by the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) server for faster restore.

Using MySQL Enterprise Backup is the recommended method for backing up **InnoDB** data.

**Note**

**InnoDB** does not support databases that are restored using third-party backup tools.

#### Hot Backups

The **mysqlbackup** command, part of the MySQL Enterprise Backup component, lets you back up a running MySQL instance, including **InnoDB** tables, with minimal disruption to operations while producing a consistent snapshot of the database. When **mysqlbackup** is copying **InnoDB** tables, reads and writes to **InnoDB** tables can continue. MySQL Enterprise Backup can also create compressed backup files, and back up subsets of tables and databases. In conjunction with the MySQL binary log, users can perform point-in-time recovery. MySQL Enterprise Backup is part of the MySQL Enterprise subscription. For more details, see [Section 30.2, “MySQL Enterprise Backup Overview”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup).

#### Cold Backups

If you can shut down the MySQL server, you can make a physical backup that consists of all files used by **InnoDB** to manage its tables. Use the following procedure:

Perform a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown) of the MySQL server and make sure that it stops without errors.

Copy all **InnoDB** data files (ibdata files and .ibd files) into a safe place.

Copy all **InnoDB** log files (ib\_logfile files) to a safe place.

Copy your my.cnf configuration file or files to a safe place.

#### Logical Backups Using mysqldump

In addition to physical backups, it is recommended that you regularly create logical backups by dumping your tables using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump). A binary file might be corrupted without you noticing it. Dumped tables are stored into text files that are human-readable, so spotting table corruption becomes easier. Also, because the format is simpler, the chance for serious data corruption is smaller. [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) also has a [--single-transaction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_single-transaction) option for making a consistent snapshot without locking out other clients. See [Section 7.3.1, “Establishing a Backup Policy”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html#backup-policy).

Replication works with [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, so you can use MySQL replication capabilities to keep a copy of your database at database sites requiring high availability. See [Section 15.19, “InnoDB and MySQL Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-and-mysql-replication).

### 15.18.2 InnoDB Recovery

This section describes **InnoDB** recovery. Topics include:

[Point-in-Time Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-point-in-time)

[Recovery from Data Corruption or Disk Failure](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-corruption-disk-failure-recovery)

[InnoDB Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-crash-recovery)

[Tablespace Discovery During Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-tablespace-discovery)

#### Point-in-Time Recovery

To recover an **InnoDB** database to the present from the time at which the physical backup was made, you must run MySQL server with binary logging enabled, even before taking the backup. To achieve point-in-time recovery after restoring a backup, you can apply changes from the binary log that occurred after the backup was made. See [Section 7.5, “Point-in-Time (Incremental) Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html#point-in-time-recovery).

#### Recovery from Data Corruption or Disk Failure

If your database becomes corrupted or disk failure occurs, you must perform the recovery using a backup. In the case of corruption, first find a backup that is not corrupted. After restoring the base backup, do a point-in-time recovery from the binary log files using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) and [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) to restore the changes that occurred after the backup was made.

In some cases of database corruption, it is enough to dump, drop, and re-create one or a few corrupt tables. You can use the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) statement to check whether a table is corrupt, although [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) naturally cannot detect every possible kind of corruption.

In some cases, apparent database page corruption is actually due to the operating system corrupting its own file cache, and the data on disk may be okay. It is best to try restarting the computer first. Doing so may eliminate errors that appeared to be database page corruption. If MySQL still has trouble starting because of **InnoDB** consistency problems, see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery) for steps to start the instance in recovery mode, which permits you to dump the data.

#### InnoDB Crash Recovery

To recover from an unexpected MySQL server exit, the only requirement is to restart the MySQL server. **InnoDB** automatically checks the logs and performs a roll-forward of the database to the present. **InnoDB** automatically rolls back uncommitted transactions that were present at the time of the crash. During recovery, [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) displays output similar to this:

InnoDB: The log sequence number 664050266 in the system tablespace does not match

the log sequence number 685111586 in the ib\_logfiles!

InnoDB: Database was not shutdown normally!

InnoDB: Starting crash recovery.

InnoDB: Using 'tablespaces.open.2' max LSN: 664075228

InnoDB: Doing recovery: scanned up to log sequence number 690354176

InnoDB: Doing recovery: scanned up to log sequence number 695597056

InnoDB: Doing recovery: scanned up to log sequence number 700839936

InnoDB: Doing recovery: scanned up to log sequence number 706082816

InnoDB: Doing recovery: scanned up to log sequence number 711325696

InnoDB: Doing recovery: scanned up to log sequence number 713458156

InnoDB: Applying a batch of 1467 redo log records ...

InnoDB: 10%

InnoDB: 20%

InnoDB: 30%

InnoDB: 40%

InnoDB: 50%

InnoDB: 60%

InnoDB: 70%

InnoDB: 80%

InnoDB: 90%

InnoDB: 100%

InnoDB: Apply batch completed!

InnoDB: 1 transaction(s) which must be rolled back or cleaned up in total 561887 row

operations to undo

InnoDB: Trx id counter is 4096

...

InnoDB: 8.0.1 started; log sequence number 713458156

InnoDB: Waiting for purge to start

InnoDB: Starting in background the rollback of uncommitted transactions

InnoDB: Rolling back trx with id 3596, 561887 rows to undo

...

./mysqld: ready for connections....

**InnoDB** [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery) consists of several steps:

Tablespace discovery

Tablespace discovery is the process that **InnoDB** uses to identify tablespaces that require redo log application. See [Tablespace Discovery During Crash Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-recovery-tablespace-discovery).

[Redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) application

Redo log application is performed during initialization, before accepting any connections. If all changes are flushed from the [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) to the [tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_tablespace) (ibdata\* and \*.ibd files) at the time of the shutdown or crash, redo log application is skipped. **InnoDB** also skips redo log application if redo log files are missing at startup.

The current maximum auto-increment counter value is written to the redo log each time the value changes, which makes it crash-safe. During recovery, **InnoDB** scans the redo log to collect counter value changes and applies the changes to the in-memory table object.

For more information about how **InnoDB** handles auto-increment values, see [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling), and [InnoDB AUTO\_INCREMENT Counter Initialization](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-initialization).

When encountering index tree corruption, **InnoDB** writes a corruption flag to the redo log, which makes the corruption flag crash-safe. **InnoDB** also writes in-memory corruption flag data to an engine-private system table on each checkpoint. During recovery, **InnoDB** reads corruption flags from both locations and merges results before marking in-memory table and index objects as corrupt.

Removing redo logs to speed up recovery is not recommended, even if some data loss is acceptable. Removing redo logs should only be considered after a clean shutdown, with [**innodb\_fast\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_fast_shutdown) set to **0** or **1**.

[Roll back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback) of incomplete [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction)

Incomplete transactions are any transactions that were active at the time of unexpected exit or [fast shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_fast_shutdown). The time it takes to roll back an incomplete transaction can be three or four times the amount of time a transaction is active before it is interrupted, depending on server load.

You cannot cancel transactions that are being rolled back. In extreme cases, when rolling back transactions is expected to take an exceptionally long time, it may be faster to start **InnoDB** with an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) setting of **3** or greater. See [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

[Change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer) merge

Applying changes from the change buffer (part of the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace)) to leaf pages of secondary indexes, as the index pages are read to the buffer pool.

[Purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge)

Deleting delete-marked records that are no longer visible to active transactions.

The steps that follow redo log application do not depend on the redo log (other than for logging the writes) and are performed in parallel with normal processing. Of these, only rollback of incomplete transactions is special to crash recovery. The insert buffer merge and the purge are performed during normal processing.

After redo log application, **InnoDB** attempts to accept connections as early as possible, to reduce downtime. As part of crash recovery, **InnoDB** rolls back transactions that were not committed or in **XA PREPARE** state when the server exited. The rollback is performed by a background thread, executed in parallel with transactions from new connections. Until the rollback operation is completed, new connections may encounter locking conflicts with recovered transactions.

In most situations, even if the MySQL server was killed unexpectedly in the middle of heavy activity, the recovery process happens automatically and no action is required of the DBA. If a hardware failure or severe system error corrupted **InnoDB** data, MySQL might refuse to start. In this case, see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

For information about the binary log and **InnoDB** crash recovery, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

#### Tablespace Discovery During Crash Recovery

If, during recovery, **InnoDB** encounters redo logs written since the last checkpoint, the redo logs must be applied to affected tablespaces. The process that identifies affected tablespaces during recovery is referred to as tablespace discovery.

Tablespace discovery relies on the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting, which defines the directories to scan at startup for tablespace files. The [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) default setting is NULL, but the directories defined by [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory), and [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir) are always appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) argument value when InnoDb builds a list of directories to scan at startup. These directories are appended regardless of whether an [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting is specified explicitly. Tablespace files defined with an absolute path or that reside outside of the directories appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting should be added to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting. Recovery is terminated if any tablespace file referenced in a redo log has not been discovered previously.

## 15.19 InnoDB and MySQL Replication

It is possible to use replication in a way where the storage engine on the replica is not the same as the storage engine on the source. For example, you can replicate modifications to an **InnoDB** table on the source to a **MyISAM** table on the replica. For more information see, [Section 17.4.4, “Using Replication with Different Source and Replica Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-diffengines).

For information about setting up a replica, see [Section 17.1.2.6, “Setting Up Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-setup-replicas), and [Section 17.1.2.5, “Choosing a Method for Data Snapshots”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-snapshot-method). To make a new replica without taking down the source or an existing replica, use the [MySQL Enterprise Backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) product.

Transactions that fail on the source do not affect replication. MySQL replication is based on the binary log where MySQL writes SQL statements that modify data. A transaction that fails (for example, because of a foreign key violation, or because it is rolled back) is not written to the binary log, so it is not sent to replicas. See [Section 13.3.1, “START TRANSACTION, COMMIT, and ROLLBACK Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

**Replication and CASCADE.** Cascading actions for **InnoDB** tables on the source are replicated on the replica only if the tables sharing the foreign key relation use **InnoDB** on both the source and replica. This is true whether you are using statement-based or row-based replication. Suppose that you have started replication, and then create two tables on the source, where **InnoDB** is defined as the default storage engine, using the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements:

CREATE TABLE fc1 (

i INT PRIMARY KEY,

j INT

);

CREATE TABLE fc2 (

m INT PRIMARY KEY,

n INT,

FOREIGN KEY ni (n) REFERENCES fc1 (i)

ON DELETE CASCADE

);

If the replica has **MyISAM** defined as the default storage engine, the same tables are created on the replica, but they use the **MyISAM** storage engine, and the **FOREIGN KEY** option is ignored. Now we insert some rows into the tables on the source:

source> **INSERT INTO fc1 VALUES (1, 1), (2, 2);**

Query OK, 2 rows affected (0.09 sec)

Records: 2 Duplicates: 0 Warnings: 0

source> **INSERT INTO fc2 VALUES (1, 1), (2, 2), (3, 1);**

Query OK, 3 rows affected (0.19 sec)

Records: 3 Duplicates: 0 Warnings: 0

At this point, on both the source and the replica, table **fc1** contains 2 rows, and table **fc2** contains 3 rows, as shown here:

source> **SELECT \* FROM fc1;**

+---+------+

| i | j |

+---+------+

| 1 | 1 |

| 2 | 2 |

+---+------+

2 rows in set (0.00 sec)

source> **SELECT \* FROM fc2;**

+---+------+

| m | n |

+---+------+

| 1 | 1 |

| 2 | 2 |

| 3 | 1 |

+---+------+

3 rows in set (0.00 sec)

replica> **SELECT \* FROM fc1;**

+---+------+

| i | j |

+---+------+

| 1 | 1 |

| 2 | 2 |

+---+------+

2 rows in set (0.00 sec)

replica> **SELECT \* FROM fc2;**

+---+------+

| m | n |

+---+------+

| 1 | 1 |

| 2 | 2 |

| 3 | 1 |

+---+------+

3 rows in set (0.00 sec)

Now suppose that you perform the following [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement on the source:

source> **DELETE FROM fc1 WHERE i=1;**

Query OK, 1 row affected (0.09 sec)

Due to the cascade, table **fc2** on the source now contains only 1 row:

source> **SELECT \* FROM fc2;**

+---+---+

| m | n |

+---+---+

| 2 | 2 |

+---+---+

1 row in set (0.00 sec)

However, the cascade does not propagate on the replica because on the replica the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) for **fc1** deletes no rows from **fc2**. The replica's copy of **fc2** still contains all of the rows that were originally inserted:

replica> **SELECT \* FROM fc2;**

+---+---+

| m | n |

+---+---+

| 1 | 1 |

| 3 | 1 |

| 2 | 2 |

+---+---+

3 rows in set (0.00 sec)

This difference is due to the fact that the cascading deletes are handled internally by the **InnoDB** storage engine, which means that none of the changes are logged.

## 15.20 InnoDB memcached Plugin

[15.20.1 Benefits of the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-benefits)

[15.20.2 InnoDB memcached Architecture](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-intro)

[15.20.3 Setting Up the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup)

[15.20.4 InnoDB memcached Multiple get and Range Query Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-multiple-get-range-query)

[15.20.5 Security Considerations for the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-security)

[15.20.6 Writing Applications for the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-developing)

[15.20.7 The InnoDB memcached Plugin and Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-replication)

[15.20.8 InnoDB memcached Plugin Internals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-internals)

[15.20.9 Troubleshooting the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-troubleshoot)

**Note**

The **InnoDB** **memcached** plugin is deprecated as of MySQL 8.0.22; expect support for it to be removed in a future version of MySQL.

The **InnoDB** **memcached** plugin (**daemon\_memcached**) provides an integrated **memcached** daemon that automatically stores and retrieves data from **InnoDB** tables, turning the MySQL server into a fast “key-value store”. Instead of formulating queries in SQL, you can use simple **get**, **set**, and **incr** operations that avoid the performance overhead associated with SQL parsing and constructing a query optimization plan. You can also access the same **InnoDB** tables through SQL for convenience, complex queries, bulk operations, and other strengths of traditional database software.

This “NoSQL-style” interface uses the **memcached** API to speed up database operations, letting **InnoDB** handle memory caching using its [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool) mechanism. Data modified through **memcached** operations such as **add**, **set**, and **incr** are stored to disk, in **InnoDB** tables. The combination of **memcached** simplicity and **InnoDB** reliability and consistency provides users with the best of both worlds, as explained in [Section 15.20.1, “Benefits of the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-benefits). For an architectural overview, see [Section 15.20.2, “InnoDB memcached Architecture”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-intro).

### 15.20.1 Benefits of the InnoDB memcached Plugin

This section outlines advantages the **daemon\_memcached** plugin. The combination of **InnoDB** tables and **memcached** offers advantages over using either by themselves.

Direct access to the **InnoDB** storage engine avoids the parsing and planning overhead of SQL.

Running **memcached** in the same process space as the MySQL server avoids the network overhead of passing requests back and forth.

Data written using the **memcached** protocol is transparently written to an **InnoDB** table, without going through the MySQL SQL layer. You can control frequency of writes to achieve higher raw performance when updating non-critical data.

Data requested through the **memcached** protocol is transparently queried from an **InnoDB** table, without going through the MySQL SQL layer.

Subsequent requests for the same data is served from the **InnoDB** buffer pool. The buffer pool handles the in-memory caching. You can tune performance of data-intensive operations using **InnoDB** configuration options.

Data can be unstructured or structured, depending on the type of application. You can create a new table for data, or use existing tables.

**InnoDB** can handle composing and decomposing multiple column values into a single **memcached** item value, reducing the amount of string parsing and concatenation required in your application. For example, you can store the string value **2|4|6|8** in the **memcached** cache, and have **InnoDB** split the value based on a separator character, then store the result in four numeric columns.

The transfer between memory and disk is handled automatically, simplifying application logic.

Data is stored in a MySQL database to protect against crashes, outages, and corruption.

You can access the underlying **InnoDB** table through SQL for reporting, analysis, ad hoc queries, bulk loading, multi-step transactional computations, set operations such as union and intersection, and other operations suited to the expressiveness and flexibility of SQL.

You can ensure high availability by using the **daemon\_memcached** plugin on a source server in combination with MySQL replication.

The integration of **memcached** with MySQL provides a way to make in-memory data persistent, so you can use it for more significant kinds of data. You can use more **add**, **incr**, and similar write operations in your application without concern that data could be lost. You can stop and start the **memcached** server without losing updates made to cached data. To guard against unexpected outages, you can take advantage of **InnoDB** crash recovery, replication, and backup capabilities.

The way **InnoDB** does fast [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) lookups is a natural fit for **memcached** single-item queries. The direct, low-level database access path used by the **daemon\_memcached** plugin is much more efficient for key-value lookups than equivalent SQL queries.

The serialization features of **memcached**, which can turn complex data structures, binary files, or even code blocks into storeable strings, offer a simple way to get such objects into a database.

Because you can access the underlying data through SQL, you can produce reports, search or update across multiple keys, and call functions such as **AVG()** and **MAX()** on **memcached** data. All of these operations are expensive or complicated using **memcached** by itself.

You do not need to manually load data into **memcached** at startup. As particular keys are requested by an application, values are retrieved from the database automatically, and cached in memory using the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool).

Because **memcached** consumes relatively little CPU, and its memory footprint is easy to control, it can run comfortably alongside a MySQL instance on the same system.

Because data consistency is enforced by mechanisms used for regular **InnoDB** tables, you do not have to worry about stale **memcached** data or fallback logic to query the database in the case of a missing key.

### 15.20.2 InnoDB memcached Architecture

The **InnoDB** **memcached** plugin implements **memcached** as a MySQL plugin daemon that accesses the **InnoDB** storage engine directly, bypassing the MySQL SQL layer.

The following diagram illustrates how an application accesses data through the **daemon\_memcached** plugin, compared with SQL.

**Figure 15.4 MySQL Server with Integrated memcached Server**

Features of the **daemon\_memcached** plugin:

**memcached** as a daemon plugin of [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld). Both [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) and **memcached** run in the same process space, with very low latency access to data.

Direct access to **InnoDB** tables, bypassing the SQL parser, the optimizer, and even the Handler API layer.

Standard **memcached** protocols, including the text-based protocol and the binary protocol. The **daemon\_memcached** plugin passes all 55 compatibility tests of the **memcapable** command.

Multi-column support. You can map multiple columns into the “value” part of the key-value store, with column values delimited by a user-specified separator character.

By default, the **memcached** protocol is used to read and write data directly to **InnoDB**, letting MySQL manage in-memory caching using the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool). The default settings represent a combination of high reliability and the fewest surprises for database applications. For example, default settings avoid uncommitted data on the database side, or stale data returned for **memcached** **get** requests.

Advanced users can configure the system as a traditional **memcached** server, with all data cached only in the **memcached** engine (memory caching), or use a combination of the “**memcached** engine” (memory caching) and the **InnoDB** **memcached** engine (**InnoDB** as back-end persistent storage).

Control over how often data is passed back and forth between **InnoDB** and **memcached** operations through the [**innodb\_api\_bk\_commit\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_bk_commit_interval), [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size), and [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) configuration options. Batch size options default to a value of 1 for maximum reliability.

The ability to specify **memcached** options through the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter. For example, you can change the port that **memcached** listens on, reduce the maximum number of simultaneous connections, change the maximum memory size for a key-value pair, or enable debugging messages for the error log.

The [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) configuration option controls the transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) on queries processed by **memcached**. Although **memcached** has no concept of [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction), you can use this option to control how soon **memcached** sees changes caused by SQL statements issued on the table used by the **daemon\_memcached** plugin. By default, [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) is set to [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted).

The [**innodb\_api\_enable\_mdl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_mdl) option can be used to lock the table at the MySQL level, so that the mapped table cannot be dropped or altered by [DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ddl) through the SQL interface. Without the lock, the table can be dropped from the MySQL layer, but kept in **InnoDB** storage until **memcached** or some other user stops using it. “MDL” stands for “metadata locking”.

#### Differences Between InnoDB memcached and Traditional memcached

You may already be familiar with using **memcached** with MySQL, as described in [Using MySQL with **memcached**](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached.html). This section describes how features of the integrated **InnoDB** **memcached** plugin differ from traditional **memcached**.

Installation: The **memcached** library comes with the MySQL server, making installation and setup relatively easy. Installation involves running the innodb\_memcached\_config.sql script to create a **demo\_test** table for **memcached** to use, issuing an [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement to enable the **daemon\_memcached** plugin, and adding desired **memcached** options to a MySQL configuration file or startup script. You might still install the traditional **memcached** distribution for additional utilities such as **memcp**, **memcat**, and **memcapable**.

For comparison with traditional **memcached**, see [Installing **memcached**](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-install.html).

Deployment: With traditional **memcached**, it is typical to run large numbers of low-capacity **memcached** servers. A typical deployment of the **daemon\_memcached** plugin, however, involves a smaller number of moderate or high-powered servers that are already running MySQL. The benefit of this configuration is in improving efficiency of individual database servers rather than exploiting unused memory or distributing lookups across large numbers of servers. In the default configuration, very little memory is used for **memcached**, and in-memory lookups are served from the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), which automatically caches the most recently and frequently used data. As with a traditional MySQL server instance, keep the value of the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) configuration option as high as practical (without causing paging at the OS level), so that as much work as possible is performed in memory.

For comparison with traditional **memcached**, see [**memcached** Deployment](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-deployment.html).

Expiry: By default (that is, using the **innodb\_only** caching policy), the latest data from the **InnoDB** table is always returned, so the expiry options have no practical effect. If you change the caching policy to **caching** or **cache\_only**, the expiry options work as usual, but requested data might be stale if it is updated in the underlying table before it expires from the memory cache.

For comparison with traditional **memcached**, see [Data Expiry](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-expiry.html).

Namespaces: **memcached** is like a large directory where you give files elaborate names with prefixes and suffixes to keep the files from conflicting. The **daemon\_memcached** plugin lets you use similar naming conventions for keys, with one addition. Key names in the format **@@*table\_id*.*key***.***table\_id*** are decoded to reference a specific a table, using mapping data from the **innodb\_memcache.containers** table. The ***key*** is looked up in or written to the specified table.

The **@@** notation only works for individual calls to **get**, **add**, and **set** functions, but not others such as **incr** or **delete**. To designate a default table for subsequent **memcached** operations within a session, perform a **get** request using the **@@** notation with a ***table\_id***, but without the key portion. For example:

get @@***table\_id***

Subsequent **get**, **set**, **incr**, **delete**, and other operations use the table designated by ***table\_id*** in the **innodb\_memcache.containers.name** column.

For comparison with traditional **memcached**, see [Using Namespaces](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-namespaces.html).

Hashing and distribution: The default configuration, which uses the **innodb\_only** caching policy, is suitable for a traditional deployment configuration where all data is available on all servers, such as a set of replica servers.

If you physically divide data, as in a sharded configuration, you can split data across several machines running the **daemon\_memcached** plugin, and use the traditional **memcached** hashing mechanism to route requests to a particular machine. On the MySQL side, you would typically let all data be inserted by **add** requests to **memcached** so that appropriate values are stored in the database on the appropriate server.

For comparison with traditional **memcached**, see [**memcached** Hashing/Distribution Types](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-hashtypes.html).

Memory usage: By default (with the **innodb\_only** caching policy), the **memcached** protocol passes information back and forth with **InnoDB** tables, and the **InnoDB** buffer pool handles in-memory lookups instead of **memcached** memory usage growing and shrinking. Relatively little memory is used on the **memcached** side.

If you switch the caching policy to **caching** or **cache\_only**, the normal rules of **memcached** memory usage apply. Memory for **memcached** data values is allocated in terms of “slabs”. You can control slab size and maximum memory used for **memcached**.

Either way, you can monitor and troubleshoot the **daemon\_memcached** plugin using the familiar [statistics](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-stats.html) system, accessed through the standard protocol, over a **telnet** session, for example. Extra utilities are not included with the **daemon\_memcached** plugin. You can use the [**memcached-tool** script](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-stats-memcached-tool.html) to install a full **memcached** distribution.

For comparison with traditional **memcached**, see [Memory Allocation within **memcached**](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-memory.html).

Thread usage: MySQL threads and **memcached** threads co-exist on the same server. Limits imposed on threads by the operating system apply to the total number of threads.

For comparison with traditional **memcached**, see [**memcached** Thread Support](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-threads.html).

Log usage: Because the **memcached** daemon is run alongside the MySQL server and writes to **stderr**, the **-v**, **-vv**, and **-vvv** options for logging write output to the MySQL [error log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_error_log).

For comparison with traditional **memcached**, see [**memcached** Logs](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-using-logs.html).

**memcached** operations: Familiar **memcached** operations such as **get**, **set**, **add**, and **delete** are available. Serialization (that is, the exact string format representing complex data structures) depends on the language interface.

For comparison with traditional **memcached**, see [Basic **memcached** Operations](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-operations.html).

Using **memcached** as a MySQL front end: This is the primary purpose of the **InnoDB** **memcached** plugin. An integrated **memcached** daemon improves application performance, and having **InnoDB** handle data transfers between memory and disk simplifies application logic.

For comparison with traditional **memcached**, see [Using **memcached** as a MySQL Caching Layer](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-mysql-frontend.html).

Utilities: The MySQL server includes the **libmemcached** library but not additional command-line utilities. To use commands such as **memcp**, **memcat**, and **memcapable** commands, install a full **memcached** distribution. When **memrm** and **memflush** remove items from the cache, the items are also removed from the underlying **InnoDB** table.

For comparison with traditional **memcached**, see [**libmemcached** Command-Line Utilities](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-libmemcached.html#ha-memcached-interfaces-libmemcached-utilities).

Programming interfaces: You can access the MySQL server through the **daemon\_memcached** plugin using all supported languages: [C and C++](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-libmemcached.html), [Java](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-java.html), [Perl](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-perl.html), [Python](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-python.html), [PHP](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-php.html), and [Ruby](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-ruby.html). Specify the server hostname and port as with a traditional **memcached** server. By default, the **daemon\_memcached** plugin listens on port **11211**. You can use both the [text and binary protocols](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-protocol.html). You can customize the [behavior](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-libmemcached.html#ha-memcached-interfaces-libmemcached-behaviors) of **memcached** functions at runtime. Serialization (that is, the exact string format representing complex data structures) depends on the language interface.

For comparison with traditional **memcached**, see [Developing a **memcached** Application](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces.html).

Frequently asked questions: MySQL has an extensive FAQ for traditional **memcached**. The FAQ is mostly applicable, except that using **InnoDB** tables as a storage medium for **memcached** data means that you can use **memcached** for more write-intensive applications than before, rather than as a read-only cache.

See [**memcached** FAQ](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-faq.html).

### 15.20.3 Setting Up the InnoDB memcached Plugin

This section describes how to set up the **daemon\_memcached** plugin on a MySQL server. Because the **memcached** daemon is tightly integrated with the MySQL server to avoid network traffic and minimize latency, you perform this process on each MySQL instance that uses this feature.

**Note**

Before setting up the **daemon\_memcached** plugin, consult [Section 15.20.5, “Security Considerations for the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-security) to understand the security procedures required to prevent unauthorized access.

#### Prerequisites

The **daemon\_memcached** plugin is only supported on Linux, Solaris, and macOS platforms. Other operating systems are not supported.

When building MySQL from source, you must build with [-DWITH\_INNODB\_MEMCACHED=ON](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_innodb_memcached). This build option generates two shared libraries in the MySQL plugin directory ([**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir)) that are required to run the **daemon\_memcached** plugin:

libmemcached.so: the **memcached** daemon plugin to MySQL.

innodb\_engine.so: an **InnoDB** API plugin to **memcached**.

**libevent** must be installed.

If you did not build MySQL from source, the **libevent** library is not included in your installation. Use the installation method for your operating system to install **libevent** 1.4.12 or later. For example, depending on the operating system, you might use **apt-get**, **yum**, or **port install**. For example, on Ubuntu Linux, use:

sudo apt-get install libevent-dev

If you installed MySQL from a source code release, **libevent** 1.4.12 is bundled with the package and is located at the top level of the MySQL source code directory. If you use the bundled version of **libevent**, no action is required. If you want to use a local system version of **libevent**, you must build MySQL with the [-DWITH\_LIBEVENT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_libevent) build option set to **system** or **yes**.

#### Installing and Configuring the InnoDB memcached Plugin

Configure the **daemon\_memcached** plugin so it can interact with **InnoDB** tables by running the innodb\_memcached\_config.sql configuration script, which is located in ***MYSQL\_HOME***/share. This script installs the **innodb\_memcache** database with three required tables (**cache\_policies**, **config\_options**, and **containers**). It also installs the **demo\_test** sample table in the **test** database.

mysql> **source *MYSQL\_HOME*/share/innodb\_memcached\_config.sql**

Running the innodb\_memcached\_config.sql script is a one-time operation. The tables remain in place if you later uninstall and re-install the **daemon\_memcached** plugin.

mysql> **USE innodb\_memcache;**

mysql> **SHOW TABLES;**

+---------------------------+

| Tables\_in\_innodb\_memcache |

+---------------------------+

| cache\_policies |

| config\_options |

| containers |

+---------------------------+

mysql> **USE test;**

mysql> **SHOW TABLES;**

+----------------+

| Tables\_in\_test |

+----------------+

| demo\_test |

+----------------+

Of these tables, the **innodb\_memcache.containers** table is the most important. Entries in the **containers** table provide a mapping to **InnoDB** table columns. Each **InnoDB** table used with the **daemon\_memcached** plugin requires an entry in the **containers** table.

The innodb\_memcached\_config.sql script inserts a single entry in the **containers** table that provides a mapping for the **demo\_test** table. It also inserts a single row of data into the **demo\_test** table. This data allows you to immediately verify the installation after the setup is completed.

mysql> **SELECT \* FROM innodb\_memcache.containers\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: aaa

db\_schema: test

db\_table: demo\_test

key\_columns: c1

value\_columns: c2

flags: c3

cas\_column: c4

expire\_time\_column: c5

unique\_idx\_name\_on\_key: PRIMARY

mysql> **SELECT \* FROM test.demo\_test;**

+----+------------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+----+------------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

+----+------------------+------+------+------+

For more information about **innodb\_memcache** tables and the **demo\_test** sample table, see [Section 15.20.8, “InnoDB memcached Plugin Internals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-internals).

Activate the **daemon\_memcached** plugin by running the [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement:

mysql> **INSTALL PLUGIN daemon\_memcached soname "libmemcached.so";**

Once the plugin is installed, it is automatically activated each time the MySQL server is restarted.

#### Verifying the InnoDB and memcached Setup

To verify the **daemon\_memcached** plugin setup, use a **telnet** session to issue **memcached** commands. By default, the **memcached** daemon listens on port 11211.

Retrieve data from the **test.demo\_test** table. The single row of data in the **demo\_test** table has a key value of **AA**.

**telnet localhost 11211**

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

**get AA**

VALUE AA 8 12

HELLO, HELLO

END

Insert data using a **set** command.

**set BB 10 0 16**

**GOODBYE, GOODBYE**

STORED

where:

**set** is the command to store a value

**BB** is the key

**10** is a flag for the operation; ignored by **memcached** but may be used by the client to indicate any type of information; specify **0** if unused

**0** is the expiration time (TTL); specify **0** if unused

**16** is the length of the supplied value block in bytes

**GOODBYE, GOODBYE** is the value that is stored

Verify that the data inserted is stored in MySQL by connecting to the MySQL server and querying the **test.demo\_test** table.

mysql> **SELECT \* FROM test.demo\_test;**

+----+------------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+----+------------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

| BB | GOODBYE, GOODBYE | 10 | 1 | 0 |

+----+------------------+------+------+------+

Return to the telnet session and retrieve the data that you inserted earlier using key **BB**.

**get BB**

VALUE BB 10 16

GOODBYE, GOODBYE

END

**quit**

If you shut down the MySQL server, which also shuts off the integrated **memcached** server, further attempts to access the **memcached** data fail with a connection error. Normally, the **memcached** data also disappears at this point, and you would require application logic to load the data back into memory when **memcached** is restarted. However, the **InnoDB** **memcached** plugin automates this process for you.

When you restart MySQL, **get** operations once again return the key-value pairs you stored in the earlier **memcached** session. When a key is requested and the associated value is not already in the memory cache, the value is automatically queried from the MySQL **test.demo\_test** table.

#### Creating a New Table and Column Mapping

This example shows how to setup your own **InnoDB** table with the **daemon\_memcached** plugin.

Create an **InnoDB** table. The table must have a key column with a unique index. The key column of the city table is **city\_id**, which is defined as the primary key. The table must also include columns for **flags**, **cas**, and **expiry** values. There may be one or more value columns. The **city** table has three value columns (**name**, **state**, **country**).

**Note**

There is no special requirement with respect to column names as along as a valid mapping is added to the **innodb\_memcache.containers** table.

mysql> **CREATE TABLE city (**

**city\_id VARCHAR(32),**

**name VARCHAR(1024),**

**state VARCHAR(1024),**

**country VARCHAR(1024),**

**flags INT,**

**cas BIGINT UNSIGNED,**

**expiry INT,**

**primary key(city\_id)**

) **ENGINE=InnoDB;**

Add an entry to the **innodb\_memcache.containers** table so that the **daemon\_memcached** plugin knows how to access the **InnoDB** table. The entry must satisfy the **innodb\_memcache.containers** table definition. For a description of each field, see [Section 15.20.8, “InnoDB memcached Plugin Internals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-internals).

mysql> **DESCRIBE innodb\_memcache.containers;**

+------------------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+------------------------+--------------+------+-----+---------+-------+

| name | varchar(50) | NO | PRI | NULL | |

| db\_schema | varchar(250) | NO | | NULL | |

| db\_table | varchar(250) | NO | | NULL | |

| key\_columns | varchar(250) | NO | | NULL | |

| value\_columns | varchar(250) | YES | | NULL | |

| flags | varchar(250) | NO | | 0 | |

| cas\_column | varchar(250) | YES | | NULL | |

| expire\_time\_column | varchar(250) | YES | | NULL | |

| unique\_idx\_name\_on\_key | varchar(250) | NO | | NULL | |

+------------------------+--------------+------+-----+---------+-------+

The **innodb\_memcache.containers** table entry for the city table is defined as:

mysql> **INSERT INTO `innodb\_memcache`.`containers` (**

**`name`, `db\_schema`, `db\_table`, `key\_columns`, `value\_columns`,**

**`flags`, `cas\_column`, `expire\_time\_column`, `unique\_idx\_name\_on\_key`)**

**VALUES ('default', 'test', 'city', 'city\_id', 'name|state|country',**

**'flags','cas','expiry','PRIMARY');**

**default** is specified for the **containers.name** column to configure the **city** table as the default **InnoDB** table to be used with the **daemon\_memcached** plugin.

Multiple **InnoDB** table columns (**name**, **state**, **country**) are mapped to **containers.value\_columns** using a “|” delimiter.

The **flags**, **cas\_column**, and **expire\_time\_column** fields of the **innodb\_memcache.containers** table are typically not significant in applications using the **daemon\_memcached** plugin. However, a designated **InnoDB** table column is required for each. When inserting data, specify **0** for these columns if they are unused.

After updating the **innodb\_memcache.containers** table, restart the **daemon\_memcache** plugin to apply the changes.

mysql> **UNINSTALL PLUGIN daemon\_memcached;**

mysql> **INSTALL PLUGIN daemon\_memcached soname "libmemcached.so";**

Using telnet, insert data into the **city** table using a **memcached** **set** command.

**telnet localhost 11211**

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

**set B 0 0 22**

**BANGALORE|BANGALORE|IN**

STORED

Using MySQL, query the **test.city** table to verify that the data you inserted was stored.

mysql> **SELECT \* FROM test.city;**

+---------+-----------+-----------+---------+-------+------+--------+

| city\_id | name | state | country | flags | cas | expiry |

+---------+-----------+-----------+---------+-------+------+--------+

| B | BANGALORE | BANGALORE | IN | 0 | 3 | 0 |

+---------+-----------+-----------+---------+-------+------+--------+

Using MySQL, insert additional data into the **test.city** table.

mysql> **INSERT INTO city VALUES ('C','CHENNAI','TAMIL NADU','IN', 0, 0 ,0);**

mysql> **INSERT INTO city VALUES ('D','DELHI','DELHI','IN', 0, 0, 0);**

mysql> **INSERT INTO city VALUES ('H','HYDERABAD','TELANGANA','IN', 0, 0, 0);**

mysql> **INSERT INTO city VALUES ('M','MUMBAI','MAHARASHTRA','IN', 0, 0, 0);**

**Note**

It is recommended that you specify a value of **0** for the **flags**, **cas\_column**, and **expire\_time\_column** fields if they are unused.

Using telnet, issue a **memcached** **get** command to retrieve data you inserted using MySQL.

**get H**

VALUE H 0 22

HYDERABAD|TELANGANA|IN

END

#### Configuring the InnoDB memcached Plugin

Traditional **memcached** configuration options may be specified in a MySQL configuration file or a [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) startup string, encoded in the argument of the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter. **memcached** configuration options take effect when the plugin is loaded, which occurs each time the MySQL server is started.

For example, to make **memcached** listen on port 11222 instead of the default port 11211, specify **-p11222** as an argument of the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration option:

mysqld .... --daemon\_memcached\_option="-p11222"

Other **memcached** options can be encoded in the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) string. For example, you can specify options to reduce the maximum number of simultaneous connections, change the maximum memory size for a key-value pair, or enable debugging messages for the error log, and so on.

There are also configuration options specific to the **daemon\_memcached** plugin. These include:

[**daemon\_memcached\_engine\_lib\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_name): Specifies the shared library that implements the **InnoDB** **memcached** plugin. The default setting is innodb\_engine.so.

[**daemon\_memcached\_engine\_lib\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_path): The path of the directory containing the shared library that implements the **InnoDB** **memcached** plugin. The default is NULL, representing the plugin directory.

[**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size): Defines the batch commit size for read operations (**get**). It specifies the number of **memcached** read operations after which a [commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit) occurs. [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) is set to 1 by default so that every **get** request accesses the most recently committed data in the **InnoDB** table, whether the data was updated through **memcached** or by SQL. When the value is greater than 1, the counter for read operations is incremented with each **get** call. A **flush\_all** call resets both read and write counters.

[**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size): Defines the batch commit size for write operations (**set**, **replace**, **append**, **prepend**, **incr**, **decr**, and so on). [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) is set to 1 by default so that no uncommitted data is lost in case of an outage, and so that SQL queries on the underlying table access the most recent data. When the value is greater than 1, the counter for write operations is incremented for each **add**, **set**, **incr**, **decr**, and **delete** call. A **flush\_all** call resets both read and write counters.

By default, you do not need to modify [**daemon\_memcached\_engine\_lib\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_name) or [**daemon\_memcached\_engine\_lib\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_engine_lib_path). You might configure these options if, for example, you want to use a different storage engine for **memcached** (such as the NDB **memcached** engine).

**daemon\_memcached** plugin configuration parameters may be specified in the MySQL configuration file or in a [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) startup string. They take effect when you load the **daemon\_memcached** plugin.

When making changes to **daemon\_memcached** plugin configuration, reload the plugin to apply the changes. To do so, issue the following statements:

mysql> **UNINSTALL PLUGIN daemon\_memcached;**

mysql> **INSTALL PLUGIN daemon\_memcached soname "libmemcached.so";**

Configuration settings, required tables, and data are preserved when the plugin is restarted.

For additional information about enabling and disabling plugins, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

### 15.20.4 InnoDB memcached Multiple get and Range Query Support

The **daemon\_memcached** plugin supports multiple get operations (fetching multiple key-value pairs in a single **memcached** query) and range queries.

#### Multiple get Operations

The ability to fetch multiple key-value pairs in a single **memcached** query improves read performance by reducing communication traffic between the client and server. For **InnoDB**, it means fewer transactions and open-table operations.

The following example demonstrates multiple-get support. The example uses the **test.city** table described in [Creating a New Table and Column Mapping](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-new-table-setup).

mysql> **USE test;**

mysql> **SELECT \* FROM test.city;**

+---------+-----------+-------------+---------+-------+------+--------+

| city\_id | name | state | country | flags | cas | expiry |

+---------+-----------+-------------+---------+-------+------+--------+

| B | BANGALORE | BANGALORE | IN | 0 | 1 | 0 |

| C | CHENNAI | TAMIL NADU | IN | 0 | 0 | 0 |

| D | DELHI | DELHI | IN | 0 | 0 | 0 |

| H | HYDERABAD | TELANGANA | IN | 0 | 0 | 0 |

| M | MUMBAI | MAHARASHTRA | IN | 0 | 0 | 0 |

+---------+-----------+-------------+---------+-------+------+--------+

Run a **get** command to retrieve all values from the **city** table. The results are returned in a key-value pair sequence.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**get B C D H M**

VALUE B 0 22

BANGALORE|BANGALORE|IN

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

VALUE M 0 21

MUMBAI|MAHARASHTRA|IN

END

When retrieving multiple values in a single **get** command, you can switch tables (using **@@*containers.name*** notation) to retrieve the value for the first key, but you cannot switch tables for subsequent keys. For example, the table switch in this example is valid:

**get @@aaa.AA BB**

VALUE @@aaa.AA 8 12

HELLO, HELLO

VALUE BB 10 16

GOODBYE, GOODBYE

END

Attempting to switch tables again in the same **get** command to retrieve a key value from a different table is not supported.

There is no limit the number of keys that can be retrieved by a multiple get operation, but there is a 128MB memory limit for storing the result.

#### Range Queries

For range queries, the **daemon\_memcached** plugin supports the following comparison operators: **<**, **>**, **<=**, **>=**. An operator must be preceded by an **@** symbol. When a range query finds multiple matching key-value pairs, results are returned in a key-value pair sequence.

The following examples demonstrate range query support. The examples use the **test.city** table described in [Creating a New Table and Column Mapping](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-new-table-setup).

mysql> **SELECT \* FROM test.city;**

+---------+-----------+-------------+---------+-------+------+--------+

| city\_id | name | state | country | flags | cas | expiry |

+---------+-----------+-------------+---------+-------+------+--------+

| B | BANGALORE | BANGALORE | IN | 0 | 1 | 0 |

| C | CHENNAI | TAMIL NADU | IN | 0 | 0 | 0 |

| D | DELHI | DELHI | IN | 0 | 0 | 0 |

| H | HYDERABAD | TELANGANA | IN | 0 | 0 | 0 |

| M | MUMBAI | MAHARASHTRA | IN | 0 | 0 | 0 |

+---------+-----------+-------------+---------+-------+------+--------+

Open a telnet session:

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

To get all values greater than **B**, enter **get @>B**:

**get @>B**

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

VALUE M 0 21

MUMBAI|MAHARASHTRA|IN

END

To get all values less than **M**, enter **get @<M**:

**get @<M**

VALUE B 0 22

BANGALORE|BANGALORE|IN

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

END

To get all values less than and including **M**, enter **get @<=M**:

**get @<=M**

VALUE B 0 22

BANGALORE|BANGALORE|IN

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

VALUE M 0 21

MUMBAI|MAHARASHTRA|IN

To get values greater than **B** but less than **M**, enter **get @>B@<M**:

**get @>B@<M**

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

END

A maximum of two comparison operators can be parsed, one being either a 'less than' (**@<**) or 'less than or equal to' (**@<=**) operator, and the other being either a 'greater than' (**@>**) or 'greater than or equal to' (**@>=**) operator. Any additional operators are assumed to be part of the key. For example, if you issue a **get** command with three operators, the third operator (**@>C**) is treated as part of the key, and the **get** command searches for values smaller than **M** and greater than **B@>C**.

**get @<M@>B@>C**

VALUE C 0 21

CHENNAI|TAMIL NADU|IN

VALUE D 0 14

DELHI|DELHI|IN

VALUE H 0 22

HYDERABAD|TELANGANA|IN

### 15.20.5 Security Considerations for the InnoDB memcached Plugin

**Caution**

Consult this section before deploying the **daemon\_memcached** plugin on a production server, or even on a test server if the MySQL instance contains sensitive data.

Because **memcached** does not use an authentication mechanism by default, and the optional SASL authentication is not as strong as traditional DBMS security measures, only keep non-sensitive data in the MySQL instance that uses the **daemon\_memcached** plugin, and wall off any servers that use this configuration from potential intruders. Do not allow **memcached** access to these servers from the Internet; only allow access from within a firewalled intranet, ideally from a subnet whose membership you can restrict.

#### Password-Protecting memcached Using SASL

SASL support provides the capability to protect your MySQL database from unauthenticated access through **memcached** clients. This section explains how to enable SASL with the **daemon\_memcached** plugin. The steps are almost identical to those performed to enabled SASL for a traditional **memcached** server.

SASL stands for “Simple Authentication and Security Layer”, a standard for adding authentication support to connection-based protocols. **memcached** added SASL support in version 1.4.3.

SASL authentication is only supported with the binary protocol.

**memcached** clients are only able to access **InnoDB** tables that are registered in the **innodb\_memcache.containers** table. Even though a DBA can place access restrictions on such tables, access through **memcached** applications cannot be controlled. For this reason, SASL support is provided to control access to **InnoDB** tables associated with the **daemon\_memcached** plugin.

The following section shows how to build, enable, and test an SASL-enabled **daemon\_memcached** plugin.

#### Building and Enabling SASL with the InnoDB memcached Plugin

By default, an SASL-enabled **daemon\_memcached** plugin is not included in MySQL release packages, since an SASL-enabled **daemon\_memcached** plugin requires building **memcached** with SASL libraries. To enable SASL support, download the MySQL source and rebuild the **daemon\_memcached** plugin after downloading the SASL libraries:

Install the SASL development and utility libraries. For example, on Ubuntu, use **apt-get** to obtain the libraries:

sudo apt-get -f install libsasl2-2 sasl2-bin libsasl2-2 libsasl2-dev libsasl2-modules

Build the **daemon\_memcached** plugin shared libraries with SASL capability by adding **ENABLE\_MEMCACHED\_SASL=1** to your **cmake** options. **memcached** also provides simple cleartext password support, which facilitates testing. To enable simple cleartext password support, specify the **ENABLE\_MEMCACHED\_SASL\_PWDB=1** **cmake** option.

In summary, add following three **cmake** options:

cmake ... -DWITH\_INNODB\_MEMCACHED=1 -DENABLE\_MEMCACHED\_SASL=1 -DENABLE\_MEMCACHED\_SASL\_PWDB=1

Install the **daemon\_memcached** plugin, as described in [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

Configure a user name and password file. (This example uses **memcached** simple cleartext password support.)

In a file, create a user named **testname** and define the password as **testpasswd**:

echo "testname:testpasswd:::::::" >/home/jy/memcached-sasl-db

Configure the **MEMCACHED\_SASL\_PWDB** environment variable to inform **memcached** of the user name and password file:

export MEMCACHED\_SASL\_PWDB=/home/jy/memcached-sasl-db

Inform **memcached** that a cleartext password is used:

echo "mech\_list: plain" > /home/jy/work2/msasl/clients/memcached.conf

export SASL\_CONF\_PATH=/home/jy/work2/msasl/clients

Enable SASL by restarting the MySQL server with the **memcached** **-S** option encoded in the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter:

mysqld ... --daemon\_memcached\_option="-S"

To test the setup, use an SASL-enabled client such as [SASL-enabled libmemcached](https://code.launchpad.net/~trond-norbye/libmemcached/sasl).

memcp --servers=localhost:11211 --binary  --username=testname

--password=***password*** myfile.txt

memcat --servers=localhost:11211 --binary --username=testname

--password=***password*** myfile.txt

If you specify an incorrect user name or password, the operation is rejected with a **memcache error AUTHENTICATION FAILURE** message. In this case, examine the cleartext password set in the memcached-sasl-db file to verify that the credentials you supplied are correct.

There are other methods to test SASL authentication with **memcached**, but the method described above is the most straightforward.

### 15.20.6 Writing Applications for the InnoDB memcached Plugin

[15.20.6.1 Adapting an Existing MySQL Schema for the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-porting-mysql)

[15.20.6.2 Adapting a memcached Application for the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-porting-memcached)

[15.20.6.3 Tuning InnoDB memcached Plugin Performance](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-tuning)

[15.20.6.4 Controlling Transactional Behavior of the InnoDB memcached Plugin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-txn)

[15.20.6.5 Adapting DML Statements to memcached Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-dml)

[15.20.6.6 Performing DML and DDL Statements on the Underlying InnoDB Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-ddl)

Typically, writing an application for the **InnoDB** **memcached** plugin involves some degree of rewriting or adapting existing code that uses MySQL or the **memcached** API.

With the **daemon\_memcached** plugin, instead of many traditional **memcached** servers running on low-powered machines, you have the same number of **memcached** servers as MySQL servers, running on relatively high-powered machines with substantial disk storage and memory. You might reuse some existing code that works with the **memcached** API, but adaptation is likely required due to the different server configuration.

The data stored through the **daemon\_memcached** plugin goes into [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, and must be converted to do numeric operations. You can perform the conversion on the application side, or by using the **CAST()** function in queries.

Coming from a database background, you might be used to general-purpose SQL tables with many columns. The tables accessed by **memcached** code likely have only a few or even a single column holding data values.

You might adapt parts of your application that perform single-row queries, inserts, updates, or deletes, to improve performance in critical sections of code. Both [queries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query) (read) and [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) (write) operations can be substantially faster when performed through the **InnoDB** **memcached** interface. The performance improvement for writes is typically greater than the performance improvement for reads, so you might focus on adapting code that performs logging or records interactive choices on a website.

The following sections explore these points in more detail.

#### 15.20.6.1 Adapting an Existing MySQL Schema for the InnoDB memcached Plugin

Consider these aspects of **memcached** applications when adapting an existing MySQL schema or application to use the **daemon\_memcached** plugin:

**memcached** keys cannot contain spaces or newlines, because these characters are used as separators in the ASCII protocol. If you are using lookup values that contain spaces, transform or hash them into values without spaces before using them as keys in calls to **add()**, **set()**, **get()**, and so on. Although theoretically these characters are allowed in keys in programs that use the binary protocol, you should restrict the characters used in keys to ensure compatibility with a broad range of clients.

If there is a short numeric [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) column in an **InnoDB** table, use it as the unique lookup key for **memcached** by converting the integer to a string value. If the **memcached** server is used for multiple applications, or with more than one **InnoDB** table, consider modifying the name to ensure that it is unique. For example, prepend the table name, or the database name and the table name, before the numeric value.

**Note**

The **daemon\_memcached** plugin supports inserts and reads on mapped **InnoDB** tables that have an **INTEGER** defined as the primary key.

You cannot use a partitioned table for data queried or stored using **memcached**.

The **memcached** protocol passes numeric values around as strings. To store numeric values in the underlying **InnoDB** table, to implement counters that can be used in SQL functions such as **SUM()** or **AVG()**, for example:

Use [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns with enough characters to hold all the digits of the largest expected number (and additional characters if appropriate for the negative sign, decimal point, or both).

In any query that performs arithmetic using column values, use the **CAST()** function to convert the values from string to integer, or to some other numeric type. For example:

# Alphabetic entries are returned as zero.

SELECT CAST(c2 as unsigned integer) FROM demo\_test;

# Since there could be numeric values of 0, can't disqualify them.

# Test the string values to find the ones that are integers, and average only those.

SELECT AVG(cast(c2 as unsigned integer)) FROM demo\_test

WHERE c2 BETWEEN '0' and '9999999999';

# Views let you hide the complexity of queries. The results are already converted;

# no need to repeat conversion functions and WHERE clauses each time.

CREATE VIEW numbers AS SELECT c1 KEY, CAST(c2 AS UNSIGNED INTEGER) val

FROM demo\_test WHERE c2 BETWEEN '0' and '9999999999';

SELECT SUM(val) FROM numbers;

**Note**

Any alphabetic values in the result set are converted into 0 by the call to **CAST()**. When using functions such as **AVG()**, which depend on the number of rows in the result set, include **WHERE** clauses to filter out non-numeric values.

If the **InnoDB** column used as a key could have values longer than 250 bytes, hash the value to less than 250 bytes.

To use an existing table with the **daemon\_memcached** plugin, define an entry for it in the **innodb\_memcache.containers** table. To make that table the default for all **memcached** requests, specify a value of **default** in the **name** column, then restart the MySQL server to make the change take effect. If you use multiple tables for different classes of **memcached** data, set up multiple entries in the **innodb\_memcache.containers** table with **name** values of your choice, then issue a **memcached** request in the form of **get @@*name*** or **set @@*name*** within the application to specify the table to be used for subsequent **memcached** requests.

For an example of using a table other than the predefined **test.demo\_test** table, see [Example 15.13, “Using Your Own Table with an InnoDB memcached Application”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-tutorial-python). For the required table layout, see [Section 15.20.8, “InnoDB memcached Plugin Internals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-internals).

To use multiple **InnoDB** table column values with **memcached** key-value pairs, specify column names separated by comma, semicolon, space, or pipe characters in the **value\_columns** field of the **innodb\_memcache.containers** entry for the **InnoDB** table. For example, specify **col1,col2,col3** or **col1|col2|col3** in the **value\_columns** field.

Concatenate the column values into a single string using the pipe character as a separator before passing the string to **memcached** **add** or **set** calls. The string is unpacked automatically into the correct column. Each **get** call returns a single string containing the column values that is also delimited by the pipe character. You can unpack the values using the appropriate application language syntax.

**Example 15.13 Using Your Own Table with an InnoDB memcached Application**

This example shows how to use your own table with a sample Python application that uses **memcached** for data manipulation.

The example assumes that the **daemon\_memcached** plugin is installed as described in [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup). It also assumes that your system is configured to run a Python script that uses the **python-memcache** module.

Create the **multicol** table which stores country information including population, area, and driver side data (**'R'** for right and **'L'** for left).

mysql> **USE test;**

mysql> **CREATE TABLE `multicol` (**

**`country` varchar(128) NOT NULL DEFAULT '',**

**`population` varchar(10) DEFAULT NULL,**

**`area\_sq\_km` varchar(9) DEFAULT NULL,**

**`drive\_side` varchar(1) DEFAULT NULL,**

**`c3` int(11) DEFAULT NULL,**

**`c4` bigint(20) unsigned DEFAULT NULL,**

**`c5` int(11) DEFAULT NULL,**

**PRIMARY KEY (`country`)**

**) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;**

Insert a record into the **innodb\_memcache.containers** table so that the **daemon\_memcached** plugin can access the **multicol** table.

mysql> **INSERT INTO innodb\_memcache.containers**

**(name,db\_schema,db\_table,key\_columns,value\_columns,flags,cas\_column,**

**expire\_time\_column,unique\_idx\_name\_on\_key)**

**VALUES**

**('bbb','test','multicol','country','population,area\_sq\_km,drive\_side',**

**'c3','c4','c5','PRIMARY');**

mysql> **COMMIT;**

The **innodb\_memcache.containers** record for the **multicol** table specifies a **name** value of **'bbb'**, which is the table identifier.

**Note**

If a single **InnoDB** table is used for all **memcached** applications, the **name** value can be set to **default** to avoid using **@@** notation to switch tables.

The **db\_schema** column is set to **test**, which is the name of the database where the **multicol** table resides.

The **db\_table** column is set to **multicol**, which is the name of the **InnoDB** table.

**key\_columns** is set to the unique **country** column. The **country** column is defined as the primary key in the **multicol** table definition.

Rather than a single **InnoDB** table column to hold a composite data value, data is divided among three table columns (**population**, **area\_sq\_km**, and **drive\_side**). To accommodate multiple value columns, a comma-separated list of columns is specified in the **value\_columns** field. The columns defined in the **value\_columns** field are the columns used when storing or retrieving values.

Values for the **flags**, **expire\_time**, and **cas\_column** fields are based on values used in the **demo.test** sample table. These fields are typically not significant in applications that use the **daemon\_memcached** plugin because MySQL keeps data synchronized, and there is no need to worry about data expiring or becoming stale.

The **unique\_idx\_name\_on\_key** field is set to **PRIMARY**, which refers to the primary index defined on the unique **country** column in the **multicol** table.

Copy the sample Python application into a file. In this example, the sample script is copied to a file named multicol.py.

The sample Python application inserts data into the **multicol** table and retrieves data for all keys, demonstrating how to access an **InnoDB** table through the **daemon\_memcached** plugin.

import sys, os

import memcache

def connect\_to\_memcached():

memc = memcache.Client(['127.0.0.1:11211'], debug=0);

print "Connected to memcached."

return memc

def banner(message):

print

print "=" \* len(message)

print message

print "=" \* len(message)

country\_data = [

("Canada","34820000","9984670","R"),

("USA","314242000","9826675","R"),

("Ireland","6399152","84421","L"),

("UK","62262000","243610","L"),

("Mexico","113910608","1972550","R"),

("Denmark","5543453","43094","R"),

("Norway","5002942","385252","R"),

("UAE","8264070","83600","R"),

("India","1210193422","3287263","L"),

("China","1347350000","9640821","R"),

]

def switch\_table(memc,table):

key = "@@" + table

print "Switching default table to '" + table + "' by issuing GET for '" + key + "'."

result = memc.get(key)

def insert\_country\_data(memc):

banner("Inserting initial data via memcached interface")

for item in country\_data:

country = item[0]

population = item[1]

area = item[2]

drive\_side = item[3]

key = country

value = "|".join([population,area,drive\_side])

print "Key = " + key

print "Value = " + value

if memc.add(key,value):

print "Added new key, value pair."

else:

print "Updating value for existing key."

memc.set(key,value)

def query\_country\_data(memc):

banner("Retrieving data for all keys (country names)")

for item in country\_data:

key = item[0]

result = memc.get(key)

print "Here is the result retrieved from the database for key " + key + ":"

print result

(m\_population, m\_area, m\_drive\_side) = result.split("|")

print "Unpacked population value: " + m\_population

print "Unpacked area value : " + m\_area

print "Unpacked drive side value: " + m\_drive\_side

if \_\_name\_\_ == '\_\_main\_\_':

memc = connect\_to\_memcached()

switch\_table(memc,"bbb")

insert\_country\_data(memc)

query\_country\_data(memc)

sys.exit(0)

Sample Python application notes:

No database authorization is required to run the application, since data manipulation is performed through the **memcached** interface. The only required information is the port number on the local system where the **memcached** daemon listens.

To make sure the application uses the **multicol** table, the **switch\_table()** function is called, which performs a dummy **get** or **set** request using **@@** notation. The **name** value in the request is **bbb**, which is the **multicol** table identifier defined in the **innodb\_memcache.containers.name** field.

A more descriptive **name** value might be used in a real-world application. This example simply illustrates that a table identifier is specified rather than the table name in **get @@...** requests.

The utility functions used to insert and query data demonstrate how to turn a Python data structure into pipe-separated values for sending data to MySQL with **add** or **set** requests, and how to unpack the pipe-separated values returned by **get** requests. This extra processing is only required when mapping a single **memcached** value to multiple MySQL table columns.

Run the sample Python application.

shell> **python multicol.py**

If successful, the sample application returns this output:

Connected to memcached.

Switching default table to 'bbb' by issuing GET for '@@bbb'.

==============================================

Inserting initial data via memcached interface

==============================================

Key = Canada

Value = 34820000|9984670|R

Added new key, value pair.

Key = USA

Value = 314242000|9826675|R

Added new key, value pair.

Key = Ireland

Value = 6399152|84421|L

Added new key, value pair.

Key = UK

Value = 62262000|243610|L

Added new key, value pair.

Key = Mexico

Value = 113910608|1972550|R

Added new key, value pair.

Key = Denmark

Value = 5543453|43094|R

Added new key, value pair.

Key = Norway

Value = 5002942|385252|R

Added new key, value pair.

Key = UAE

Value = 8264070|83600|R

Added new key, value pair.

Key = India

Value = 1210193422|3287263|L

Added new key, value pair.

Key = China

Value = 1347350000|9640821|R

Added new key, value pair.

============================================

Retrieving data for all keys (country names)

============================================

Here is the result retrieved from the database for key Canada:

34820000|9984670|R

Unpacked population value: 34820000

Unpacked area value : 9984670

Unpacked drive side value: R

Here is the result retrieved from the database for key USA:

314242000|9826675|R

Unpacked population value: 314242000

Unpacked area value : 9826675

Unpacked drive side value: R

Here is the result retrieved from the database for key Ireland:

6399152|84421|L

Unpacked population value: 6399152

Unpacked area value : 84421

Unpacked drive side value: L

Here is the result retrieved from the database for key UK:

62262000|243610|L

Unpacked population value: 62262000

Unpacked area value : 243610

Unpacked drive side value: L

Here is the result retrieved from the database for key Mexico:

113910608|1972550|R

Unpacked population value: 113910608

Unpacked area value : 1972550

Unpacked drive side value: R

Here is the result retrieved from the database for key Denmark:

5543453|43094|R

Unpacked population value: 5543453

Unpacked area value : 43094

Unpacked drive side value: R

Here is the result retrieved from the database for key Norway:

5002942|385252|R

Unpacked population value: 5002942

Unpacked area value : 385252

Unpacked drive side value: R

Here is the result retrieved from the database for key UAE:

8264070|83600|R

Unpacked population value: 8264070

Unpacked area value : 83600

Unpacked drive side value: R

Here is the result retrieved from the database for key India:

1210193422|3287263|L

Unpacked population value: 1210193422

Unpacked area value : 3287263

Unpacked drive side value: L

Here is the result retrieved from the database for key China:

1347350000|9640821|R

Unpacked population value: 1347350000

Unpacked area value : 9640821

Unpacked drive side value: R

Query the **innodb\_memcache.containers** table to view the record you inserted earlier for the **multicol** table. The first record is the sample entry for the **demo\_test** table that is created during the initial **daemon\_memcached** plugin setup. The second record is the entry you inserted for the **multicol** table.

mysql> **SELECT \* FROM innodb\_memcache.containers\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: aaa

db\_schema: test

db\_table: demo\_test

key\_columns: c1

value\_columns: c2

flags: c3

cas\_column: c4

expire\_time\_column: c5

unique\_idx\_name\_on\_key: PRIMARY

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: bbb

db\_schema: test

db\_table: multicol

key\_columns: country

value\_columns: population,area\_sq\_km,drive\_side

flags: c3

cas\_column: c4

expire\_time\_column: c5

unique\_idx\_name\_on\_key: PRIMARY

Query the **multicol** table to view data inserted by the sample Python application. The data is available for MySQL [queries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_query), which demonstrates how the same data can be accessed using SQL or through applications (using the appropriate [MySQL Connector or API](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\connectors-apis.html)).

mysql> **SELECT \* FROM test.multicol;**

+---------+------------+------------+------------+------+------+------+

| country | population | area\_sq\_km | drive\_side | c3 | c4 | c5 |

+---------+------------+------------+------------+------+------+------+

| Canada | 34820000 | 9984670 | R | 0 | 11 | 0 |

| China | 1347350000 | 9640821 | R | 0 | 20 | 0 |

| Denmark | 5543453 | 43094 | R | 0 | 16 | 0 |

| India | 1210193422 | 3287263 | L | 0 | 19 | 0 |

| Ireland | 6399152 | 84421 | L | 0 | 13 | 0 |

| Mexico | 113910608 | 1972550 | R | 0 | 15 | 0 |

| Norway | 5002942 | 385252 | R | 0 | 17 | 0 |

| UAE | 8264070 | 83600 | R | 0 | 18 | 0 |

| UK | 62262000 | 243610 | L | 0 | 14 | 0 |

| USA | 314242000 | 9826675 | R | 0 | 12 | 0 |

+---------+------------+------------+------------+------+------+------+

**Note**

Always allow sufficient size to hold necessary digits, decimal points, sign characters, leading zeros, and so on when defining the length for columns that are treated as numbers. Too-long values in a string column such as a **VARCHAR** are truncated by removing some characters, which could produce nonsensical numeric values.

Optionally, run report-type queries on the **InnoDB** table that stores the **memcached** data.

You can produce reports through SQL queries, performing calculations and tests across any columns, not just the **country** key column. (Because the following examples use data from only a few countries, the numbers are for illustration purposes only.) The following queries return the average population of countries where people drive on the right, and the average size of countries whose names start with “U”:

mysql> **SELECT AVG(population) FROM multicol WHERE drive\_side = 'R';**

+-------------------+

| avg(population) |

+-------------------+

| 261304724.7142857 |

+-------------------+

mysql> **SELECT SUM(area\_sq\_km) FROM multicol WHERE country LIKE 'U%';**

+-----------------+

| sum(area\_sq\_km) |

+-----------------+

| 10153885 |

+-----------------+

Because the **population** and **area\_sq\_km** columns store character data rather than strongly typed numeric data, functions such as **AVG()** and **SUM()** work by converting each value to a number first. This approach does not work for operators such as **<** or **>**, for example, when comparing character-based values, **9 > 1000**, which is not expected from a clause such as **ORDER BY population DESC**. For the most accurate type treatment, perform queries against views that cast numeric columns to the appropriate types. This technique lets you issue simple **SELECT \*** queries from database applications, while ensuring that casting, filtering, and ordering is correct. The following example shows a view that can be queried to find the top three countries in descending order of population, with the results reflecting the latest data in the **multicol** table, and with population and area figures treated as numbers:

mysql> **CREATE VIEW populous\_countries AS**

**SELECT**

**country,**

**cast(population as unsigned integer) population,**

**cast(area\_sq\_km as unsigned integer) area\_sq\_km,**

**drive\_side FROM multicol**

**ORDER BY CAST(population as unsigned integer) DESC**

**LIMIT 3;**

mysql> **SELECT \* FROM populous\_countries;**

+---------+------------+------------+------------+

| country | population | area\_sq\_km | drive\_side |

+---------+------------+------------+------------+

| China | 1347350000 | 9640821 | R |

| India | 1210193422 | 3287263 | L |

| USA | 314242000 | 9826675 | R |

+---------+------------+------------+------------+

mysql> **DESC populous\_countries;**

+------------+---------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+------------+---------------------+------+-----+---------+-------+

| country | varchar(128) | NO | | | |

| population | bigint(10) unsigned | YES | | NULL | |

| area\_sq\_km | int(9) unsigned | YES | | NULL | |

| drive\_side | varchar(1) | YES | | NULL | |

+------------+---------------------+------+-----+---------+-------+

#### 15.20.6.2 Adapting a memcached Application for the InnoDB memcached Plugin

Consider these aspects of MySQL and **InnoDB** tables when adapting existing **memcached** applications to use the **daemon\_memcached** plugin:

If there are key values longer than a few bytes, it may be more efficient to use a numeric auto-increment column as the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) of the **InnoDB** table, and to create a unique [secondary index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index) on the column that contains the **memcached** key values. This is because **InnoDB** performs best for large-scale insertions if primary key values are added in sorted order (as they are with auto-increment values). Primary key values are included in secondary indexes, which takes up unnecessary space if the primary key is a long string value.

If you store several different classes of information using **memcached**, consider setting up a separate **InnoDB** table for each type of data. Define additional table identifiers in the **innodb\_memcache.containers** table, and use the **@@*table\_id*.*key*** notation to store and retrieve items from different tables. Physically dividing different types of information allows you tune the characteristics of each table for optimum space utilization, performance, and reliability. For example, you might enable [compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) for a table that holds blog posts, but not for a table that holds thumbnail images. You might back up one table more frequently than another because it holds critical data. You might create additional [secondary indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index) on tables that are frequently used to generate reports using SQL.

Preferably, configure a stable set of table definitions for use with the **daemon\_memcached** plugin, and leave the tables in place permanently. Changes to the **innodb\_memcache.containers** table take effect the next time the **innodb\_memcache.containers** table is queried. Entries in the containers table are processed at startup, and are consulted whenever an unrecognized table identifier (as defined by **containers.name**) is requested using **@@** notation. Thus, new entries are visible as soon as you use the associated table identifier, but changes to existing entries require a server restart before they take effect.

When you use the default **innodb\_only** caching policy, calls to **add()**, **set()**, **incr()**, and so on can succeed but still trigger debugging messages such as **while expecting 'STORED', got unexpected response 'NOT\_STORED**. Debug messages occur because new and updated values are sent directly to the **InnoDB** table without being saved in the memory cache, due to the **innodb\_only** caching policy.

#### 15.20.6.3 Tuning InnoDB memcached Plugin Performance

Because using **InnoDB** in combination with **memcached** involves writing all data to disk, whether immediately or sometime later, raw performance is expected to be somewhat slower than using **memcached** by itself. When using the **InnoDB** **memcached** plugin, focus tuning goals for **memcached** operations on achieving better performance than equivalent SQL operations.

Benchmarks suggest that queries and [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) operations (inserts, updates, and deletes) that use the **memcached** interface are faster than traditional SQL. DML operations typically see a larger improvements. Therefore, consider adapting write-intensive applications to use the **memcached** interface first. Also consider prioritizing adaptation of write-intensive applications that use fast, lightweight mechanisms that lack reliability.

##### Adapting SQL Queries

The types of queries that are most suited to simple **GET** requests are those with a single clause or a set of **AND** conditions in the **WHERE** clause:

SQL:

SELECT col FROM tbl WHERE key = 'key\_value';

memcached:

get key\_value

SQL:

SELECT col FROM tbl WHERE col1 = val1 and col2 = val2 and col3 = val3;

memcached:

# Since you must always know these 3 values to look up the key,

# combine them into a unique string and use that as the key

# for all ADD, SET, and GET operations.

key\_value = val1 + ":" + val2 + ":" + val3

get key\_value

SQL:

SELECT 'key exists!' FROM tbl

WHERE EXISTS (SELECT col1 FROM tbl WHERE KEY = 'key\_value') LIMIT 1;

memcached:

# Test for existence of key by asking for its value and checking if the call succeeds,

# ignoring the value itself. For existence checking, you typically only store a very

# short value such as "1".

get key\_value

##### Using System Memory

For best performance, deploy the **daemon\_memcached** plugin on machines that are configured as typical database servers, where the majority of system RAM is devoted to the **InnoDB** [buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), through the [**innodb\_buffer\_pool\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_size) configuration option. For systems with multi-gigabyte buffer pools, consider raising the value of [**innodb\_buffer\_pool\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_buffer_pool_instances) for maximum throughput when most operations involve data that is already cached in memory.

##### Reducing Redundant I/O

**InnoDB** has a number of settings that let you choose the balance between high reliability, in case of a crash, and the amount of I/O overhead during high write workloads. For example, consider setting the [**innodb\_doublewrite**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_doublewrite) to **0** and [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) to **2**. Measure performance with different [**innodb\_flush\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_method) settings.

For other ways to reduce or tune I/O for table operations, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizing-innodb-diskio).

##### Reducing Transactional Overhead

A default value of 1 for [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) and [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) is intended for maximum reliability of results and safety of stored or updated data.

Depending on the type of application, you might increase one or both of these settings to reduce the overhead of frequent [commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit) operations. On a busy system, you might increase [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size), knowing that changes to data made through SQL may not become visible to **memcached** immediately (that is, until ***N*** more **get** operations are processed). When processing data where every write operation must be reliably stored, leave [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) set to **1**. Increase the setting when processing large numbers of updates intended only for statistical analysis, where losing the last ***N*** updates in an unexpected exit is an acceptable risk.

For example, imagine a system that monitors traffic crossing a busy bridge, recording data for approximately 100,000 vehicles each day. If the application counts different types of vehicles to analyze traffic patterns, changing [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) from **1** to **100** reduces I/O overhead for commit operations by 99%. In case of an outage, a maximum of 100 records are lost, which may be an acceptable margin of error. If instead the application performed automated toll collection for each car, you would set [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) to **1** to ensure that each toll record is immediately saved to disk.

Because of the way **InnoDB** organizes **memcached** key values on disk, if you have a large number of keys to create, it may be faster to sort the data items by key value in the application and **add** them in sorted order, rather than create keys in arbitrary order.

The **memslap** command, which is part of the regular **memcached** distribution but not included with the **daemon\_memcached** plugin, can be useful for benchmarking different configurations. It can also be used to generate sample key-value pairs to use in your own benchmarks. See [**libmemcached** Command-Line Utilities](https://dev.mysql.com/doc/refman/5.6/en/ha-memcached-interfaces-libmemcached.html#ha-memcached-interfaces-libmemcached-utilities) for details.

#### 15.20.6.4 Controlling Transactional Behavior of the InnoDB memcached Plugin

Unlike traditional **memcached**, the **daemon\_memcached** plugin allows you to control durability of data values produced through calls to **add**, **set**, **incr**, and so on. By default, data written through the **memcached** interface is stored to disk, and calls to **get** return the most recent value from disk. Although the default behavior does not offer the best possible raw performance, it is still fast compared to the SQL interface for **InnoDB** tables.

As you gain experience using the **daemon\_memcached** plugin, you can consider relaxing durability settings for non-critical classes of data, at the risk of losing some updated values in the event of an outage, or returning data that is slightly out-of-date.

##### Frequency of Commits

One tradeoff between durability and raw performance is how frequently new and changed data is [committed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_commit). If data is critical, is should be committed immediately so that it is safe in case of an unexpected exit or outage. If data is less critical, such as counters that are reset after an unexpected exit or logging data that you can afford to lose, you might prefer higher raw throughput that is available with less frequent commits.

When a **memcached** operation inserts, updates, or deletes data in the underlying **InnoDB** table, the change might be committed to the **InnoDB** table instantly (if [**daemon\_memcached\_w\_batch\_size=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size)) or some time later (if the [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) value is greater than 1). In either case, the change cannot be rolled back. If you increase the value of [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) to avoid high I/O overhead during busy times, commits could become infrequent when the workload decreases. As a safety measure, a background thread automatically commits changes made through the **memcached** API at regular intervals. The interval is controlled by the [**innodb\_api\_bk\_commit\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_bk_commit_interval) configuration option, which has a default setting of **5** seconds.

When a **memcached** operation inserts or updates data in the underlying **InnoDB** table, the changed data is immediately visible to other **memcached** requests because the new value remains in the memory cache, even if it is not yet committed on the MySQL side.

##### Transaction Isolation

When a **memcached** operation such as **get** or **incr** causes a query or DML operation on the underlying **InnoDB** table, you can control whether the operation sees the very latest data written to the table, only data that has been committed, or other variations of transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level). Use the [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) configuration option to control this feature. The numeric values specified for this option correspond to isolation levels such as [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read). See the description of the [**innodb\_api\_trx\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_trx_level) option for information about other settings.

A strict isolation level ensures that data you retrieve is not rolled back or changed suddenly causing subsequent queries to return different values. However, strict isolation levels require greater [locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_locking) overhead, which can cause waits. For a NoSQL-style application that does not use long-running transactions, you can typically use the default isolation level or switch to a less strict isolation level.

##### Disabling Row Locks for memcached DML Operations

The [**innodb\_api\_disable\_rowlock**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_disable_rowlock) option can be used to disable row locks when **memcached** requests through the **daemon\_memcached** plugin cause DML operations. By default, **innodb\_api\_disable\_rowlock** is set to **OFF** which means that **memcached** requests row locks for **get** and **set** operations. When **innodb\_api\_disable\_rowlock** is set to **ON**, **memcached** requests a table lock instead of row locks.

The **innodb\_api\_disable\_rowlock** option is not dynamic. It must be specified at startup on the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) command line or entered in a MySQL configuration file.

##### Allowing or Disallowing DDL

By default, you can perform [DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ddl) operations such as [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on tables used by the **daemon\_memcached** plugin. To avoid potential slowdowns when these tables are used for high-throughput applications, disable DDL operations on these tables by enabling [**innodb\_api\_enable\_mdl**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_mdl) at startup. This option is less appropriate when accessing the same tables through both **memcached** and SQL, because it blocks [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statements on the tables, which could be important for running reporting queries.

##### Storing Data on Disk, in Memory, or Both

The **innodb\_memcache.cache\_policies** table specifies whether to store data written through the **memcached** interface to disk (**innodb\_only**, the default); in memory only, as with traditional **memcached** (**cache\_only**); or both (**caching**).

With the **caching** setting, if **memcached** cannot find a key in memory, it searches for the value in an **InnoDB** table. Values returned from **get** calls under the **caching** setting could be out-of-date if the values were updated on disk in the **InnoDB** table but are not yet expired from the memory cache.

The caching policy can be set independently for **get**, **set** (including **incr** and **decr**), **delete**, and **flush** operations.

For example, you might allow **get** and **set** operations to query or update a table and the **memcached** memory cache at the same time (using the **caching** setting), while making **delete**, **flush**, or both operate only on the in-memory copy (using the **cache\_only** setting). That way, deleting or flushing an item only expires the item from the cache, and the latest value is returned from the **InnoDB** table the next time the item is requested.

mysql> **SELECT \* FROM innodb\_memcache.cache\_policies;**

+--------------+-------------+-------------+---------------+--------------+

| policy\_name | get\_policy | set\_policy | delete\_policy | flush\_policy |

+--------------+-------------+-------------+---------------+--------------+

| cache\_policy | innodb\_only | innodb\_only | innodb\_only | innodb\_only |

+--------------+-------------+-------------+---------------+--------------+

mysql> **UPDATE innodb\_memcache.cache\_policies SET set\_policy = 'caching'**

**WHERE policy\_name = 'cache\_policy';**

**innodb\_memcache.cache\_policies** values are only read at startup. After changing values in this table, uninstall and reinstall the **daemon\_memcached** plugin to ensure that changes take effect.

mysql> **UNINSTALL PLUGIN daemon\_memcached;**

mysql> **INSTALL PLUGIN daemon\_memcached soname "libmemcached.so";**

#### 15.20.6.5 Adapting DML Statements to memcached Operations

Benchmarks suggest that the **daemon\_memcached** plugin speeds up [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) operations (inserts, updates, and deletes) more than it speeds up queries. Therefore, consider focussing initial development efforts on write-intensive applications that are I/O-bound, and look for opportunities to use MySQL with the **daemon\_memcached** plugin for new write-intensive applications.

Single-row DML statements are the easiest types of statements to turn into **memcached** operations. **INSERT** becomes **add**, **UPDATE** becomes **set**, **incr** or **decr**, and **DELETE** becomes **delete**. These operations are guaranteed to only affect one row when issued through the **memcached** interface, because the ***key*** is unique within the table.

In the following SQL examples, **t1** refers to the table used for **memcached** operations, based on the configuration in the **innodb\_memcache.containers** table. **key** refers to the column listed under **key\_columns**, and **val** refers to the column listed under **value\_columns**.

INSERT INTO t1 (key,val) VALUES (***some\_key***,***some\_value***);

SELECT val FROM t1 WHERE key = ***some\_key***;

UPDATE t1 SET val = ***new\_value*** WHERE key = ***some\_key***;

UPDATE t1 SET val = val + x WHERE key = ***some\_key***;

DELETE FROM t1 WHERE key = ***some\_key***;

The following [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements, which remove all rows from the table, correspond to the **flush\_all** operation, where **t1** is configured as the table for **memcached** operations, as in the previous example.

TRUNCATE TABLE t1;

DELETE FROM t1;

#### 15.20.6.6 Performing DML and DDL Statements on the Underlying InnoDB Table

You can access the underlying **InnoDB** table (which is **test.demo\_test** by default) through standard SQL interfaces. However, there are some restrictions:

When querying a table that is also accessed through the **memcached** interface, remember that **memcached** operations can be configured to be committed periodically rather than after every write operation. This behavior is controlled by the [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) option. If this option is set to a value greater than **1**, use [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted) queries to find rows that were just inserted.

mysql> **SET SESSSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;**

mysql> **SELECT \* FROM demo\_test;**

+------+------+------+------+-----------+------+------+------+------+------+------+

| cx   | cy   | c1   | cz   | c2        | ca   | CB   | c3   | cu   | c4   | C5   |

+------+------+------+------+-----------+------+------+------+------+------+------+

| NULL | NULL | a11  | NULL | 123456789 | NULL | NULL |   10 | NULL |    3 | NULL |

+------+------+------+------+-----------+------+------+------+------+------+------+

When modifying a table using SQL that is also accessed through the **memcached** interface, you can configure **memcached** operations to start a new transaction periodically rather than for every read operation. This behavior is controlled by the [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) option. If this option is set to a value greater than **1**, changes made to the table using SQL are not immediately visible to **memcached** operations.

The **InnoDB** table is either IS (intention shared) or IX (intention exclusive) locked for all operations in a transaction. If you increase [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) and [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) substantially from their default value of **1**, the table is most likely locked between each operation, preventing [DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ddl) statements on the table.

### 15.20.7 The InnoDB memcached Plugin and Replication

Because the **daemon\_memcached** plugin supports the MySQL [binary log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_binary_log), source server through the **memcached** interface can be replicated for backup, balancing intensive read workloads, and high availability. All **memcached** commands are supported with binary logging.

You do not need to set up the **daemon\_memcached** plugin on replica servers. The primary advantage of this configuration is increased write throughput on the source. The speed of the replication mechanism is not affected.

The following sections show how to use the binary log capability when using the **daemon\_memcached** plugin with MySQL replication. It is assumed that you have completed the setup described in [Section 15.20.3, “Setting Up the InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached-setup).

#### Enabling the InnoDB memcached Binary Log

To use the **daemon\_memcached** plugin with the MySQL [binary log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_binary_log), enable the [**innodb\_api\_enable\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_api_enable_binlog) configuration option on the source server. This option can only be set at server startup. You must also enable the MySQL binary log on the source server using the [**--log-bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) option. You can add these options to the MySQL configuration file, or on the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) command line.

mysqld ... --log-bin -–innodb\_api\_enable\_binlog=1

Configure the source and replica server, as described in [Section 17.1.2, “Setting Up Binary Log File Position Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto).

Use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to create a source data snapshot, and sync the snapshot to the replica server.

source shell> **mysqldump --all-databases --lock-all-tables > dbdump.db**

replica shell> **mysql < dbdump.db**

On the source server, issue [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) to obtain the source binary log coordinates.

mysql> **SHOW MASTER STATUS;**

On the replica server, use a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) to set up a replica server using the source binary log coordinates.

mysql> **CHANGE MASTER TO**

**MASTER\_HOST='localhost',**

**MASTER\_USER='root',**

**MASTER\_PASSWORD='',**

**MASTER\_PORT = 13000,**

**MASTER\_LOG\_FILE='0.000001,**

**MASTER\_LOG\_POS=114;**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO**

**SOURCE\_HOST='localhost',**

**SOURCE\_USER='root',**

**SOURCE\_PASSWORD='',**

**SOURCE\_PORT = 13000,**

**SOURCE\_LOG\_FILE='0.000001,**

**SOURCE\_LOG\_POS=114;**

Start the replica.

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **START REPLICA;**

If the error log prints output similar to the following, the replica is ready for replication.

2013-09-24T13:04:38.639684Z 49 [Note] Replication I/O thread: connected to

source 'root@localhost:13000', replication started in log '0.000001'

at position 114

#### Testing the InnoDB memcached Replication Configuration

This example demonstrates how to test the **InnoDB** **memcached** replication configuration using the **memcached** and telnet to insert, update, and delete data. A MySQL client is used to verify results on the source and replica servers.

The example uses the **demo\_test** table, which was created by the innodb\_memcached\_config.sql configuration script during the initial setup of the **daemon\_memcached** plugin. The **demo\_test** table contains a single example record.

Use the **set** command to insert a record with a key of **test1**, a flag value of **10**, an expiration value of **0**, a cas value of 1, and a value of **t1**.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**set test1 10 0 1**

**t1**

STORED

On the source server, check that the record was inserted into the **demo\_test** table. Assuming the **demo\_test** table was not previously modified, there should be two records. The example record with a key of **AA**, and the record you just inserted, with a key of **test1**. The **c1** column maps to the key, the **c2** column to the value, the **c3** column to the flag value, the **c4** column to the cas value, and the **c5** column to the expiration time. The expiration time was set to 0, since it is unused.

mysql> **SELECT \* FROM test.demo\_test;**

+-------+--------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+-------+--------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

| test1 | t1 | 10 | 1 | 0 |

+-------+--------------+------+------+------+

Check to verify that the same record was replicated to the replica server.

mysql> **SELECT \* FROM test.demo\_test;**

+-------+--------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+-------+--------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

| test1 | t1 | 10 | 1 | 0 |

+-------+--------------+------+------+------+

Use the **set** command to update the key to a value of **new**.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**set test1 10 0 2**

**new**

STORED

The update is replicated to the replica server (notice that the **cas** value is also updated).

mysql> **SELECT \* FROM test.demo\_test;**

+-------+--------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+-------+--------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

| test1 | new | 10 | 2 | 0 |

+-------+--------------+------+------+------+

Delete the **test1** record using a **delete** command.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**delete test1**

DELETED

When the **delete** operation is replicated to the replica, the **test1** record on the replica is also deleted.

mysql> **SELECT \* FROM test.demo\_test;**

+----+--------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+----+--------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

+----+--------------+------+------+------+

Remove all rows from the table using the **flush\_all** command.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**flush\_all**

OK

mysql> **SELECT \* FROM test.demo\_test;**

Empty set (0.00 sec)

Telnet to the source server and enter two new records.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'

**set test2 10 0 4**

**again**

STORED

**set test3 10 0 5**

**again1**

STORED

Confirm that the two records were replicated to the replica server.

mysql> **SELECT \* FROM test.demo\_test;**

+-------+--------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+-------+--------------+------+------+------+

| test2 | again | 10 | 4 | 0 |

| test3 | again1 | 10 | 5 | 0 |

+-------+--------------+------+------+------+

Remove all rows from the table using the **flush\_all** command.

**telnet 127.0.0.1 11211**

Trying 127.0.0.1...

Connected to 127.0.0.1.

Escape character is '^]'.

**flush\_all**

OK

Check to ensure that the **flush\_all** operation was replicated on the replica server.

mysql> **SELECT \* FROM test.demo\_test;**

Empty set (0.00 sec)

#### InnoDB memcached Binary Log Notes

Binary Log Format:

Most **memcached** operations are mapped to [DML](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dml) statements (analogous to insert, delete, update). Since there is no actual SQL statement being processed by the MySQL server, all **memcached** commands (except for **flush\_all**) use Row-Based Replication (RBR) logging, which is independent of any server [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) setting.

The **memcached** **flush\_all** command is mapped to the [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) command in MySQL 5.7 and earlier. Since [DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ddl) commands can only use statement-based logging, the **flush\_all** command is replicated by sending a [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement. In MySQL 8.0 and later, **flush\_all** is mapped to **DELETE** but is still replicated by sending a [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement.

Transactions:

The concept of [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction) has not typically been part of **memcached** applications. For performance considerations, [**daemon\_memcached\_r\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_r_batch_size) and [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) are used to control the batch size for read and write transactions. These settings do not affect replication. Each SQL operation on the underlying **InnoDB** table is replicated after successful completion.

The default value of [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) is **1**, which means that each **memcached** write operation is committed immediately. This default setting incurs a certain amount of performance overhead to avoid inconsistencies in the data that is visible on the source and replica servers. The replicated records are always available immediately on the replica server. If you set [**daemon\_memcached\_w\_batch\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_w_batch_size) to a value greater than **1**, records inserted or updated through **memcached** are not immediately visible on the source server; to view the records on the source server before they are committed, issue [**SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction).

### 15.20.8 InnoDB memcached Plugin Internals

#### InnoDB API for the InnoDB memcached Plugin

The **InnoDB** **memcached** engine accesses **InnoDB** through **InnoDB** APIs, most of which are directly adopted from embedded **InnoDB**. **InnoDB** API functions are passed to the **InnoDB** **memcached** engine as callback functions. **InnoDB** API functions access the **InnoDB** tables directly, and are mostly DML operations with the exception of [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table).

**memcached** commands are implemented through the **InnoDB** **memcached** API. The following table outlines how **memcached** commands are mapped to DML or DDL operations.

**Table 15.27 memcached Commands and Associated DML or DDL Operations**

| **memcached Command** | **DML or DDL Operations** |
| --- | --- |
| **get** | a read/fetch command |
| **set** | a search followed by an **INSERT** or **UPDATE** (depending on whether or not a key exists) |
| **add** | a search followed by an **INSERT** or **UPDATE** |
| **replace** | a search followed by an **UPDATE** |
| **append** | a search followed by an **UPDATE** (appends data to the result before **UPDATE**) |
| **prepend** | a search followed by an **UPDATE** (prepends data to the result before **UPDATE**) |
| **incr** | a search followed by an **UPDATE** |
| **decr** | a search followed by an **UPDATE** |
| **delete** | a search followed by a **DELETE** |
| **flush\_all** | **TRUNCATE TABLE** (DDL) |

#### InnoDB memcached Plugin Configuration Tables

This section describes configuration tables used by the **daemon\_memcached** plugin. The **cache\_policies** table, **config\_options** table, and **containers** table are created by the innodb\_memcached\_config.sql configuration script in the **innodb\_memcache** database.

mysql> **USE innodb\_memcache;**

Database changed

mysql> **SHOW TABLES;**

+---------------------------+

| Tables\_in\_innodb\_memcache |

+---------------------------+

| cache\_policies |

| config\_options |

| containers |

+---------------------------+

#### cache\_policies Table

The **cache\_policies** table defines a cache policy for the **InnoDB** **memcached** installation. You can specify individual policies for **get**, **set**, **delete**, and **flush** operations, within a single cache policy. The default setting for all operations is **innodb\_only**.

**innodb\_only**: Use **InnoDB** as the data store.

**cache\_only**: Use the **memcached** engine as the data store.

**caching**: Use both **InnoDB** and the **memcached** engine as data stores. In this case, if **memcached** cannot find a key in memory, it searches for the value in an **InnoDB** table.

**disable**: Disable caching.

**Table 15.28 cache\_policies Columns**

| **Column** | **Description** |
| --- | --- |
| **policy\_name** | Name of the cache policy. The default cache policy name is **cache\_policy**. |
| **get\_policy** | The cache policy for get operations. Valid values are **innodb\_only**, **cache\_only**, **caching**, or **disabled**. The default setting is **innodb\_only**. |
| **set\_policy** | The cache policy for set operations. Valid values are **innodb\_only**, **cache\_only**, **caching**, or **disabled**. The default setting is **innodb\_only**. |
| **delete\_policy** | The cache policy for delete operations. Valid values are **innodb\_only**, **cache\_only**, **caching**, or **disabled**. The default setting is **innodb\_only**. |
| **flush\_policy** | The cache policy for flush operations. Valid values are **innodb\_only**, **cache\_only**, **caching**, or **disabled**. The default setting is **innodb\_only**. |

#### config\_options Table

The **config\_options** table stores **memcached**-related settings that can be changed at runtime using SQL. Supported configuration options are **separator** and **table\_map\_delimiter**.

**Table 15.29 config\_options Columns**

| **Column** | **Description** |
| --- | --- |
| **Name** | Name of the **memcached**-related configuration option. The following configuration options are supported by the **config\_options** table:  **separator**: Used to separate values of a long string into separate values when there are multiple **value\_columns** defined. By default, the **separator** is a **|** character. For example, if you define **col1, col2** as value columns, and you define **|** as the separator, you can issue the following **memcached** command to insert values into **col1** and **col2**, respectively:  set keyx 10 0 19  valuecolx|valuecoly  **valuecol1x** is stored in **col1** and **valuecoly** is stored in **col2**.  **table\_map\_delimiter**: The character separating the schema name and the table name when you use the **@@** notation in a key name to access a key in a specific table. For example, **@@t1.some\_key** and **@@t2.some\_key** have the same key value, but are stored in different tables. |
| **Value** | The value assigned to the **memcached**-related configuration option. |

#### containers Table

The **containers** table is the most important of the three configuration tables. Each **InnoDB** table that is used to store **memcached** values must have an entry in the **containers** table. The entry provides a mapping between **InnoDB** table columns and container table columns, which is required for **memcached** to work with **InnoDB** tables.

The **containers** table contains a default entry for the **test.demo\_test** table, which is created by the innodb\_memcached\_config.sql configuration script. To use the **daemon\_memcached** plugin with your own **InnoDB** table, you must create an entry in the **containers** table.

**Table 15.30 containers Columns**

| **Column** | **Description** |
| --- | --- |
| **name** | The name given to the container. If an **InnoDB** table is not requested by name using **@@** notation, the **daemon\_memcached** plugin uses the **InnoDB** table with a **containers.name** value of **default**. If there is no such entry, the first entry in the **containers** table, ordered alphabetically by **name** (ascending), determines the default **InnoDB** table. |
| **db\_schema** | The name of the database where the **InnoDB** table resides. This is a required value. |
| **db\_table** | The name of the **InnoDB** table that stores **memcached** values. This is a required value. |
| **key\_columns** | The column in the **InnoDB** table that contains lookup key values for **memcached** operations. This is a required value. |
| **value\_columns** | The **InnoDB** table columns (one or more) that store **memcached** data. Multiple columns can be specified using the separator character specified in the **innodb\_memcached.config\_options** table. By default, the separator is a pipe character (“|”). To specify multiple columns, separate them with the defined separator character. For example: **col1|col2|col3**. This is a required value. |
| **flags** | The **InnoDB** table columns that are used as flags (a user-defined numeric value that is stored and retrieved along with the main value) for **memcached**. A flag value can be used as a column specifier for some operations (such as **incr**, **prepend**) if a **memcached** value is mapped to multiple columns, so that an operation is performed on a specified column. For example, if you have mapped a **value\_columns** to three **InnoDB** table columns, and only want the increment operation performed on one columns, use the **flags** column to specify the column. If you do not use the **flags** column, set a value of **0** to indicate that it is unused. |
| **cas\_column** | The **InnoDB** table column that stores compare-and-swap (cas) values. The **cas\_column** value is related to the way **memcached** hashes requests to different servers and caches data in memory. Because the **InnoDB** **memcached** plugin is tightly integrated with a single **memcached** daemon, and the in-memory caching mechanism is handled by MySQL and the [InnoDB buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), this column is rarely needed. If you do not use this column, set a value of **0** to indicate that it is unused. |
| **expire\_time\_column** | The **InnoDB** table column that stores expiration values. The **expire\_time\_column** value is related to the way **memcached** hashes requests to different servers and caches data in memory. Because the **InnoDB** **memcached** plugin is tightly integrated with a single **memcached** daemon, and the in-memory caching mechanism is handled by MySQL and the [InnoDB buffer pool](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_buffer_pool), this column is rarely needed. If you do not use this column, set a value of **0** to indicate that the column is unused. The maximum expire time is defined as **INT\_MAX32** or 2147483647 seconds (approximately 68 years). |
| **unique\_idx\_name\_on\_key** | The name of the index on the key column. It must be a unique index. It can be the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key) or a [secondary index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index). Preferably, use the primary key of the **InnoDB** table. Using the primary key avoids a lookup that is performed when using a secondary index. You cannot make a [covering index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_covering_index) for **memcached** lookups; **InnoDB** returns an error if you try to define a composite secondary index over both the key and value columns. |

##### containers Table Column Constraints

You must supply a value for **db\_schema**, **db\_name**, **key\_columns**, **value\_columns** and **unique\_idx\_name\_on\_key**. Specify **0** for **flags**, **cas\_column**, and **expire\_time\_column** if they are unused. Failing to do so could cause your setup to fail.

**key\_columns**: The maximum limit for a **memcached** key is 250 characters, which is enforced by **memcached**. The mapped key must be a non-Null [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) type.

**value\_columns**: Must be mapped to a [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column. There is no length restriction and the value can be NULL.

**cas\_column**: The **cas** value is a 64 bit integer. It must be mapped to a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) of at least 8 bytes. If you do not use this column, set a value of **0** to indicate that it is unused.

**expiration\_time\_column**: Must mapped to an [**INTEGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) of at least 4 bytes. Expiration time is defined as a 32-bit integer for Unix time (the number of seconds since January 1, 1970, as a 32-bit value), or the number of seconds starting from the current time. For the latter, the number of seconds may not exceed 60\*60\*24\*30 (the number of seconds in 30 days). If the number sent by a client is larger, the server considers it to be a real Unix time value rather than an offset from the current time. If you do not use this column, set a value of **0** to indicate that it is unused.

**flags**: Must be mapped to an [**INTEGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) of at least 32-bits and can be NULL. If you do not use this column, set a value of **0** to indicate that it is unused.

A pre-check is performed at plugin load time to enforce column constraints. If mismatches are found, the plugin is not loaded.

##### Multiple Value Column Mapping

During plugin initialization, when **InnoDB** **memcached** is configured with information defined in the **containers** table, each mapped column defined in **containers.value\_columns** is verified against the mapped **InnoDB** table. If multiple **InnoDB** table columns are mapped, there is a check to ensure that each column exists and is the right type.

At run-time, for **memcached** insert operations, if there are more delimited values than the number of mapped columns, only the number of mapped values are taken. For example, if there are six mapped columns, and seven delimited values are provided, only the first six delimited values are taken. The seventh delimited value is ignored.

If there are fewer delimited values than mapped columns, unfilled columns are set to NULL. If an unfilled column cannot be set to NULL, insert operations fail.

If a table has more columns than mapped values, the extra columns do not affect results.

#### The demo\_test Example Table

The innodb\_memcached\_config.sql configuration script creates a **demo\_test** table in the **test** database, which can be used to verify **InnoDB** **memcached** plugin installation immediately after setup.

The innodb\_memcached\_config.sql configuration script also creates an entry for the **demo\_test** table in the **innodb\_memcache.containers** table.

mysql> **SELECT \* FROM innodb\_memcache.containers\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: aaa

db\_schema: test

db\_table: demo\_test

key\_columns: c1

value\_columns: c2

flags: c3

cas\_column: c4

expire\_time\_column: c5

unique\_idx\_name\_on\_key: PRIMARY

mysql> **SELECT \* FROM test.demo\_test;**

+----+------------------+------+------+------+

| c1 | c2 | c3 | c4 | c5 |

+----+------------------+------+------+------+

| AA | HELLO, HELLO | 8 | 0 | 0 |

+----+------------------+------+------+------+

### 15.20.9 Troubleshooting the InnoDB memcached Plugin

This section describes issues that you may encounter when using the **InnoDB** **memcached** plugin.

If you encounter the following error in the MySQL error log, the server might fail to start:

failed to set rlimit for open files. Try running as root or requesting smaller maxconns value.

The error message is from the **memcached** daemon. One solution is to raise the OS limit for the number of open files. The commands for checking and increasing the open file limit varies by operating system. This example shows commands for Linux and macOS:

# Linux

shell> **ulimit -n**

1024

shell> **ulimit -n 4096**

shell> **ulimit -n**

4096

# macOS

shell> **ulimit -n**

256

shell> **ulimit -n 4096**

shell> **ulimit -n**

4096

The other solution is to reduce the number of concurrent connections permitted for the **memcached** daemon. To do so, encode the **-c** **memcached** option in the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter in the MySQL configuration file. The **-c** option has a default value of 1024.

[mysqld]

...

loose-daemon\_memcached\_option='-c 64'

To troubleshoot problems where the **memcached** daemon is unable to store or retrieve **InnoDB** table data, encode the **-vvv** **memcached** option in the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter in the MySQL configuration file. Examine the MySQL error log for debug output related to **memcached** operations.

[mysqld]

...

loose-daemon\_memcached\_option='-vvv'

If columns specified to hold **memcached** values are the wrong data type, such as a numeric type instead of a string type, attempts to store key-value pairs fail with no specific error code or message.

If the **daemon\_memcached** plugin causes MySQL server startup issues, you can temporarily disable the **daemon\_memcached** plugin while troubleshooting by adding this line under the **[mysqld]** group in the MySQL configuration file:

daemon\_memcached=OFF

For example, if you run the [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement before running the innodb\_memcached\_config.sql configuration script to set up the necessary database and tables, the server might unexpectedly exit and fail to start. The server could also fail to start if you incorrectly configure an entry in the **innodb\_memcache.containers** table.

To uninstall the **memcached** plugin for a MySQL instance, issue the following statement:

mysql> **UNINSTALL PLUGIN daemon\_memcached;**

If you run more than one instance of MySQL on the same machine with the **daemon\_memcached** plugin enabled in each instance, use the [**daemon\_memcached\_option**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_daemon_memcached_option) configuration parameter to specify a unique **memcached** port for each **daemon\_memcached** plugin.

If an SQL statement cannot find the **InnoDB** table or finds no data in the table, but **memcached** API calls retrieve the expected data, you may be missing an entry for the **InnoDB** table in the **innodb\_memcache.containers** table, or you may have not switched to the correct **InnoDB** table by issuing a **get** or **set** request using **@@*table\_id*** notation. This problem could also occur if you change an existing entry in the **innodb\_memcache.containers** table without restarting the MySQL server afterward. The free-form storage mechanism is flexible enough that your requests to store or retrieve a multi-column value such as **col1|col2|col3** may still work, even if the daemon is using the **test.demo\_test** table which stores values in a single column.

When defining your own **InnoDB** table for use with the **daemon\_memcached** plugin, and columns in the table are defined as **NOT NULL**, ensure that values are supplied for the **NOT NULL** columns when inserting a record for the table into the **innodb\_memcache.containers** table. If the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement for the **innodb\_memcache.containers** record contains fewer delimited values than there are mapped columns, unfilled columns are set to **NULL**. Attempting to insert a **NULL** value into a **NOT NULL** column causes the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) to fail, which may only become evident after you reinitialize the **daemon\_memcached** plugin to apply changes to the **innodb\_memcache.containers** table.

If **cas\_column** and **expire\_time\_column** fields of the **innodb\_memcached.containers** table are set to **NULL**, the following error is returned when attempting to load the **memcached** plugin:

InnoDB\_Memcached: column 6 in the entry for config table 'containers' in

database 'innodb\_memcache' has an invalid NULL value.

The **memcached** plugin rejects usage of **NULL** in the **cas\_column** and **expire\_time\_column** columns. Set the value of these columns to **0** when the columns are unused.

As the length of the **memcached** key and values increase, you might encounter size and length limits.

When the key exceeds 250 bytes, **memcached** operations return an error. This is currently a fixed limit within **memcached**.

**InnoDB** table limits may be encountered if values exceed 768 bytes in size, 3072 bytes in size, or half of the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. These limits primarily apply if you intend to create an index on a value column to run report-generating queries on that column using SQL. See [Section 15.22, “InnoDB Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-limits) for details.

The maximum size for the key-value combination is 1 MB.

If you share configuration files across MySQL servers of different versions, using the latest configuration options for the **daemon\_memcached** plugin could cause startup errors on older MySQL versions. To avoid compatibility problems, use the **loose** prefix with option names. For example, use **loose-daemon\_memcached\_option='-c 64'** instead of **daemon\_memcached\_option='-c 64'**.

There is no restriction or check in place to validate character set settings. **memcached** stores and retrieves keys and values in bytes and is therefore not character set sensitive. However, you must ensure that the **memcached** client and the MySQL table use the same character set.

**memcached** connections are blocked from accessing tables that contain an indexed virtual column. Accessing an indexed virtual column requires a callback to the server, but a **memcached** connection does not have access to the server code.

## 15.21 InnoDB Troubleshooting

[15.21.1 Troubleshooting InnoDB I/O Problems](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#error-creating-innodb)

[15.21.2 Forcing InnoDB Recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery)

[15.21.3 Troubleshooting InnoDB Data Dictionary Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-troubleshooting-datadict)

[15.21.4 InnoDB Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-error-handling)

The following general guidelines apply to troubleshooting **InnoDB** problems:

When an operation fails or you suspect a bug, look at the MySQL server error log (see [Section 5.4.2, “The Error Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log)). [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html) provides troubleshooting information for some of the common **InnoDB**-specific errors that you may encounter.

If the failure is related to a [deadlock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock), run with the [**innodb\_print\_all\_deadlocks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_all_deadlocks) option enabled so that details about each deadlock are printed to the MySQL server error log. For information about deadlocks, see [Section 15.7.5, “Deadlocks in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-deadlocks).

If the issue is related to the **InnoDB** data dictionary, see [Section 15.21.3, “Troubleshooting InnoDB Data Dictionary Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-troubleshooting-datadict).

When troubleshooting, it is usually best to run the MySQL server from the command prompt, rather than through [**mysqld\_safe**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld-safe) or as a Windows service. You can then see what [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) prints to the console, and so have a better grasp of what is going on. On Windows, start [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with the [--console](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_console) option to direct the output to the console window.

Enable the **InnoDB** Monitors to obtain information about a problem (see [Section 15.17, “InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-monitors)). If the problem is performance-related, or your server appears to be hung, you should enable the standard Monitor to print information about the internal state of **InnoDB**. If the problem is with locks, enable the Lock Monitor. If the problem is with table creation, tablespaces, or data dictionary operations, refer to the [InnoDB Information Schema system tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-information-schema-system-tables) to examine contents of the **InnoDB** internal data dictionary.

**InnoDB** temporarily enables standard **InnoDB** Monitor output under the following conditions:

A long semaphore wait

**InnoDB** cannot find free blocks in the buffer pool

Over 67% of the buffer pool is occupied by lock heaps or the adaptive hash index

If you suspect that a table is corrupt, run [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) on that table.

### 15.21.1 Troubleshooting InnoDB I/O Problems

The troubleshooting steps for **InnoDB** I/O problems depend on when the problem occurs: during startup of the MySQL server, or during normal operations when a DML or DDL statement fails due to problems at the file system level.

#### Initialization Problems

If something goes wrong when **InnoDB** attempts to initialize its tablespace or its log files, delete all files created by **InnoDB**: all ibdata files and all ib\_logfile files. If you already created some **InnoDB** tables, also delete any .ibd files from the MySQL database directories. Then try the **InnoDB** database creation again. For easiest troubleshooting, start the MySQL server from a command prompt so that you see what is happening.

#### Runtime Problems

If **InnoDB** prints an operating system error during a file operation, usually the problem has one of the following solutions:

Make sure the **InnoDB** data file directory and the **InnoDB** log directory exist.

Make sure [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) has access rights to create files in those directories.

Make sure [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) can read the proper my.cnf or my.ini option file, so that it starts with the options that you specified.

Make sure the disk is not full and you are not exceeding any disk quota.

Make sure that the names you specify for subdirectories and data files do not clash.

Doublecheck the syntax of the [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir) and [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) values. In particular, any **MAX** value in the [**innodb\_data\_file\_path**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_file_path) option is a hard limit, and exceeding that limit causes a fatal error.

### 15.21.2 Forcing InnoDB Recovery

To investigate database page corruption, you might dump your tables from the database with [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into). Usually, most of the data obtained in this way is intact. Serious corruption might cause **SELECT \* FROM *tbl\_name*** statements or **InnoDB** background operations to unexpectedly exit or assert, or even cause **InnoDB** roll-forward recovery to crash. In such cases, you can use the [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) option to force the **InnoDB** storage engine to start up while preventing background operations from running, so that you can dump your tables. For example, you can add the following line to the **[mysqld]** section of your option file before restarting the server:

[mysqld]

innodb\_force\_recovery = 1

For information about using option files, see [Section 4.2.2.2, “Using Option Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files).

**Warning**

Only set [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) to a value greater than 0 in an emergency situation, so that you can start **InnoDB** and dump your tables. Before doing so, ensure that you have a backup copy of your database in case you need to recreate it. Values of 4 or greater can permanently corrupt data files. Only use an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) setting of 4 or greater on a production server instance after you have successfully tested the setting on a separate physical copy of your database. When forcing **InnoDB** recovery, you should always start with [**innodb\_force\_recovery=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) and only increase the value incrementally, as necessary.

[**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) is 0 by default (normal startup without forced recovery). The permissible nonzero values for [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) are 1 to 6. A larger value includes the functionality of lesser values. For example, a value of 3 includes all of the functionality of values 1 and 2.

If you are able to dump your tables with an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) value of 3 or less, then you are relatively safe that only some data on corrupt individual pages is lost. A value of 4 or greater is considered dangerous because data files can be permanently corrupted. A value of 6 is considered drastic because database pages are left in an obsolete state, which in turn may introduce more corruption into [B-trees](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_b_tree) and other database structures.

As a safety measure, **InnoDB** prevents [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations when [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) is greater than 0. An [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) setting of 4 or greater places **InnoDB** in read-only mode.

**1** (**SRV\_FORCE\_IGNORE\_CORRUPT**)

Lets the server run even if it detects a corrupt [page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page). Tries to make **SELECT \* FROM *tbl\_name*** jump over corrupt index records and pages, which helps in dumping tables.

**2** (**SRV\_FORCE\_NO\_BACKGROUND**)

Prevents the [master thread](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_master_thread) and any [purge threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge_thread) from running. If an unexpected exit would occur during the [purge](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_purge) operation, this recovery value prevents it.

**3** (**SRV\_FORCE\_NO\_TRX\_UNDO**)

Does not run transaction [rollbacks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback) after [crash recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_crash_recovery).

**4** (**SRV\_FORCE\_NO\_IBUF\_MERGE**)

Prevents [insert buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_insert_buffer) merge operations. If they would cause a crash, does not do them. Does not calculate table [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics). This value can permanently corrupt data files. After using this value, be prepared to drop and recreate all secondary indexes. Sets **InnoDB** to read-only.

**5** (**SRV\_FORCE\_NO\_UNDO\_LOG\_SCAN**)

Does not look at [undo logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_undo_log) when starting the database: **InnoDB** treats even incomplete transactions as committed. This value can permanently corrupt data files. Sets **InnoDB** to read-only.

**6** (**SRV\_FORCE\_NO\_LOG\_REDO**)

Does not do the [redo log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redo_log) roll-forward in connection with recovery. This value can permanently corrupt data files. Leaves database pages in an obsolete state, which in turn may introduce more corruption into B-trees and other database structures. Sets **InnoDB** to read-only.

You can [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) from tables to dump them. With an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) value of 3 or less you can **DROP** or **CREATE** tables. [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) is also supported with an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) value greater than 3. [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) is not permitted with an [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) value greater than 4.

If you know that a given table is causing an unexpected exit on rollback, you can drop it. If you encounter a runaway rollback caused by a failing mass import or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), you can kill the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process and set [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) to **3** to bring the database up without the rollback, and then **DROP** the table that is causing the runaway rollback.

If corruption within the table data prevents you from dumping the entire table contents, a query with an **ORDER BY *primary\_key* DESC** clause might be able to dump the portion of the table after the corrupted part.

If a high [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) value is required to start **InnoDB**, there may be corrupted data structures that could cause complex queries (queries containing **WHERE**, **ORDER BY**, or other clauses) to fail. In this case, you may only be able to run basic **SELECT \* FROM t** queries.

### 15.21.3 Troubleshooting InnoDB Data Dictionary Operations

Information about table definitions is stored in the InnoDB [data dictionary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_data_dictionary). If you move data files around, dictionary data can become inconsistent.

If a data dictionary corruption or consistency issue prevents you from starting **InnoDB**, see [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery) for information about manual recovery.

#### Cannot Open Datafile

With [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) enabled (the default), the following messages may appear at startup if a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace file (**.ibd** file) is missing:

[ERROR] InnoDB: Operating system error number 2 in a file operation.

[ERROR] InnoDB: The error means the system cannot find the path specified.

[ERROR] InnoDB: Cannot open datafile for read-only: './test/t1.ibd' OS error: 71

[Warning] InnoDB: Ignoring tablespace `test/t1` because it could not be opened.

To address these messages, issue [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement to remove data about the missing table from the data dictionary.

#### Restoring Orphan File-Per-Table ibd Files

This procedure describes how to restore orphan [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) .ibd files to another MySQL instance. You might use this procedure if the system tablespace is lost or unrecoverable and you want to restore .ibd file backups on a new MySQL instance.

The procedure is not supported for [general tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace) .ibd files.

The procedure assumes that you only have .ibd file backups, you are recovering to the same version of MySQL that initially created the orphan .ibd files, and that .ibd file backups are clean. See [Section 15.6.1.4, “Moving or Copying InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-migration) for information about creating clean backups.

Table import limitations outlined in [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import) are applicable to this procedure.

On the new MySQL instance, recreate the table in a database of the same name.

mysql> **CREATE DATABASE sakila;**

mysql> **USE sakila;**

mysql> **CREATE TABLE actor (**

**actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,**

**first\_name VARCHAR(45) NOT NULL,**

**last\_name VARCHAR(45) NOT NULL,**

**last\_update TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**PRIMARY KEY (actor\_id),**

**KEY idx\_actor\_last\_name (last\_name)**

**)ENGINE=InnoDB DEFAULT CHARSET=utf8;**

Discard the tablespace of the newly created table.

mysql> **ALTER TABLE sakila.actor DISCARD TABLESPACE;**

Copy the orphan **.ibd** file from your backup directory to the new database directory.

shell> **cp /backup\_directory/actor.ibd *path/to/mysql-5.7/data*/sakila/**

Ensure that the .ibd file has the necessary file permissions.

Import the orphan **.ibd** file. A warning is issued indicating that **InnoDB** is attempting to import the file without schema verification.

mysql> **ALTER TABLE sakila.actor IMPORT TABLESPACE; SHOW WARNINGS;**

Query OK, 0 rows affected, 1 warning (0.15 sec)

Warning | 1810 | InnoDB: IO Read error: (2, No such file or directory)

Error opening './sakila/actor.cfg', will attempt to import

without schema verification

Query the table to verify that the **.ibd** file was successfully restored.

mysql> **SELECT COUNT(\*) FROM sakila.actor;**

+----------+

| count(\*) |

+----------+

| 200 |

+----------+

### 15.21.4 InnoDB Error Handling

The following items describe how **InnoDB** performs error handling. **InnoDB** sometimes rolls back only the statement that failed, other times it rolls back the entire transaction.

If you run out of file space in a [tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_tablespace), a MySQL **Table is full** error occurs and **InnoDB** rolls back the SQL statement.

A transaction [deadlock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_deadlock) causes **InnoDB** to [roll back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rollback) the entire [transaction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction). Retry the entire transaction when this happens.

A lock wait timeout causes **InnoDB** to roll back the current statement (the statement that was waiting for the lock and encountered the timeout). To have the entire transaction roll back, start the server with [--innodb-rollback-on-timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) enabled. Retry the statement if using the default behavior, or the entire transaction if [--innodb-rollback-on-timeout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_rollback_on_timeout) is enabled.

Both deadlocks and lock wait timeouts are normal on busy servers and it is necessary for applications to be aware that they may happen and handle them by retrying. You can make them less likely by doing as little work as possible between the first change to data during a transaction and the commit, so the locks are held for the shortest possible time and for the smallest possible number of rows. Sometimes splitting work between different transactions may be practical and helpful.

A duplicate-key error rolls back the SQL statement, if you have not specified the **IGNORE** option in your statement.

A **row too long error** rolls back the SQL statement.

Other errors are mostly detected by the MySQL layer of code (above the **InnoDB** storage engine level), and they roll back the corresponding SQL statement. Locks are not released in a rollback of a single SQL statement.

During implicit rollbacks, as well as during the execution of an explicit [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) SQL statement, [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) displays **Rolling back** in the **State** column for the relevant connection.

## 15.22 InnoDB Limits

This section describes limits for **InnoDB** tables, indexes, tablespaces, and other aspects of the **InnoDB** storage engine.

A table can contain a maximum of 1017 columns. Virtual generated columns are included in this limit.

A table can contain a maximum of 64 [secondary indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_secondary_index).

The index key prefix length limit is 3072 bytes for **InnoDB** tables that use [**DYNAMIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dynamic_row_format) or [**COMPRESSED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compressed_row_format) row format.

The index key prefix length limit is 767 bytes for **InnoDB** tables that use the [**REDUNDANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redundant_row_format) or [**COMPACT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compact_row_format) row format. For example, you might hit this limit with a [column prefix](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_column_prefix) index of more than 191 characters on a **TEXT** or **VARCHAR** column, assuming a **utf8mb4** character set and the maximum of 4 bytes for each character.

Attempting to use an index key prefix length that exceeds the limit returns an error.

If you reduce the **InnoDB** [page size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page_size) to 8KB or 4KB by specifying the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) option when creating the MySQL instance, the maximum length of the index key is lowered proportionally, based on the limit of 3072 bytes for a 16KB page size. That is, the maximum index key length is 1536 bytes when the page size is 8KB, and 768 bytes when the page size is 4KB.

The limits that apply to index key prefixes also apply to full-column index keys.

A maximum of 16 columns is permitted for multicolumn indexes. Exceeding the limit returns an error.

ERROR 1070 (42000): Too many key parts specified; max 16 parts allowed

The maximum row size, excluding any variable-length columns that are stored off-page, is slightly less than half of a page for 4KB, 8KB, 16KB, and 32KB page sizes. For example, the maximum row size for the default [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) of 16KB is about 8000 bytes. However, for an **InnoDB** page size of 64KB, the maximum row size is approximately 16000 bytes. [**LONGBLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**LONGTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns must be less than 4GB, and the total row size, including [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, must be less than 4GB.

If a row is less than half a page long, all of it is stored locally within the page. If it exceeds half a page, variable-length columns are chosen for external off-page storage until the row fits within half a page, as described in [Section 15.11.2, “File Space Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-space).

Although **InnoDB** supports row sizes larger than 65,535 bytes internally, MySQL itself imposes a row-size limit of 65,535 for the combined size of all columns. See [Section 8.4.7, “Limits on Table Column Count and Row Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-count-limit).

On some older operating systems, files must be less than 2GB. This is not an **InnoDB** limitation. If you require a large system tablespace, configure it using several smaller data files rather than one large data file, or distribute table data across file-per-table and general tablespace data files.

The combined maximum size for **InnoDB** log files is 512GB.

The minimum tablespace size is slightly larger than 10MB. The maximum tablespace size depends on the **InnoDB** page size.

**Table 15.31 InnoDB Maximum Tablespace Size**

| **InnoDB Page Size** | **Maximum Tablespace Size** |
| --- | --- |
| 4KB | 16TB |
| 8KB | 32TB |
| 16KB | 64TB |
| 32KB | 128TB |
| 64KB | 256TB |

The maximum tablespace size is also the maximum size for a table.

The path of a tablespace file, including the file name, cannot exceed the **MAX\_PATH** limit on Windows. Prior to Windows 10, the **MAX\_PATH** limit is 260 characters. As of Windows 10, version 1607, **MAX\_PATH** limitations are removed from common Win32 file and directory functions, but you must enable the new behavior.

For limits associated with concurrent read-write transactions, see [Section 15.6.6, “Undo Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-logs).

## 15.23 InnoDB Restrictions and Limitations

This section describes restrictions and limitations of the **InnoDB** storage engine.

You cannot create a table with a column name that matches the name of an internal **InnoDB** column (including **DB\_ROW\_ID**, **DB\_TRX\_ID**, and **DB\_ROLL\_PTR**. This restriction applies to use of the names in any lettercase.

mysql> **CREATE TABLE t1 (c1 INT, db\_row\_id INT) ENGINE=INNODB;**

ERROR 1166 (42000): Incorrect column name 'db\_row\_id'

[**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) does not provide accurate statistics for **InnoDB** tables except for the physical size reserved by the table. The row count is only a rough estimate used in SQL optimization.

**InnoDB** does not keep an internal count of rows in a table because concurrent transactions might “see” different numbers of rows at the same time. Consequently, **SELECT COUNT(\*)** statements only count rows visible to the current transaction.

For information about how **InnoDB** processes **SELECT COUNT(\*)** statements, refer to the [**COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) description in [Section 12.20.1, “Aggregate Function Descriptions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions).

**ROW\_FORMAT=COMPRESSED** is unsupported for page sizes greater than 16KB.

A MySQL instance using a particular **InnoDB** page size ([**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size)) cannot use data files or log files from an instance that uses a different page size.

For limitations associated with importing tables using the Transportable Tablespaces feature, see [Table Import Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-statistics-estimation).

For limitations associated with online DDL, see [Section 15.12.6, “Online DDL Limitations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-limitations).

For limitations associated with general tablespaces, see [General Tablespace Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-limitations).

For limitations associated with data-at-rest encryption, see [Encryption Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption-limitations).

|  |  |  |
| --- | --- | --- |
| [Prev](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-dictionary.html) | Up | [Next](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html) |
| Chapter 14 MySQL Data Dictionary | [Home](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\index.html) | Chapter 16 Alternative Storage Engines |

# 其他存储引擎

存储引擎是处理不同表类型的SQL操作的MySQL组件。InnoDB是默认的和最通用的存储引擎，甲骨文公司建议除专门的使用情况外，对表使用它。(MySQL 8.0中的CREATE TABLE语句默认创建InnoDB表)。

MySQL服务器使用一个可插拔的存储引擎架构，使存储引擎能够被加载到运行中的MySQL服务器中并从其中卸载。

要确定你的服务器支持哪些存储引擎，请使用SHOW ENGINES语句。支持栏中的值表示是否可以使用一个引擎。一个YES、NO或DEFAULT的值表示一个引擎是可用的、不可用的或可用的，并且当前被设置为默认存储引擎。

mysql> **SHOW ENGINES\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: PERFORMANCE\_SCHEMA

Support: YES

Comment: Performance Schema

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: InnoDB

Support: DEFAULT

Comment: Supports transactions, row-level locking, and foreign keys

Transactions: YES

XA: YES

Savepoints: YES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: MRG\_MYISAM

Support: YES

Comment: Collection of identical MyISAM tables

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: BLACKHOLE

Support: YES

Comment: /dev/null storage engine (anything you write to it disappears)

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: MyISAM

Support: YES

Comment: MyISAM storage engine

Transactions: NO

XA: NO

Savepoints: NO

...

This chapter covers use cases for special-purpose MySQL storage engines. It does not cover the default [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine or the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine which are covered in [Chapter 15, *The InnoDB Storage Engine*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [Chapter 23, *MySQL NDB Cluster 8.0*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html). For advanced users, it also contains a description of the pluggable storage engine architecture (see [Section 16.11, “Overview of MySQL Storage Engine Architecture”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#pluggable-storage-overview)).

For information about features offered in commercial MySQL Server binaries, see [MySQL Editions](https://www.mysql.com/products/), on the MySQL website. The storage engines available might depend on which edition of MySQL you are using.

For answers to commonly asked questions about MySQL storage engines, see [Section A.2, “MySQL 8.0 FAQ: Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\faqs.html#faqs-storage-engines).

## MySQL 8.0 Supported Storage Engines

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html): The default storage engine in MySQL 8.0. **InnoDB** is a transaction-safe (ACID compliant) storage engine for MySQL that has commit, rollback, and crash-recovery capabilities to protect user data. **InnoDB** row-level locking (without escalation to coarser granularity locks) and Oracle-style consistent nonlocking reads increase multi-user concurrency and performance. **InnoDB** stores user data in clustered indexes to reduce I/O for common queries based on primary keys. To maintain data integrity, **InnoDB** also supports **FOREIGN KEY** referential-integrity constraints. For more information about **InnoDB**, see [Chapter 15, *The InnoDB Storage Engine*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html).

[**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine): These tables have a small footprint. [Table-level locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_table_lock) limits the performance in read/write workloads, so it is often used in read-only or read-mostly workloads in Web and data warehousing configurations.

[**Memory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine): Stores all data in RAM, for fast access in environments that require quick lookups of non-critical data. This engine was formerly known as the **HEAP** engine. Its use cases are decreasing; **InnoDB** with its buffer pool memory area provides a general-purpose and durable way to keep most or all data in memory, and **NDBCLUSTER** provides fast key-value lookups for huge distributed data sets.

[**CSV**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#csv-storage-engine): Its tables are really text files with comma-separated values. CSV tables let you import or dump data in CSV format, to exchange data with scripts and applications that read and write that same format. Because CSV tables are not indexed, you typically keep the data in **InnoDB** tables during normal operation, and only use CSV tables during the import or export stage.

[**Archive**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine): These compact, unindexed tables are intended for storing and retrieving large amounts of seldom-referenced historical, archived, or security audit information.

[**Blackhole**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#blackhole-storage-engine): The Blackhole storage engine accepts but does not store data, similar to the Unix **/dev/null** device. Queries always return an empty set. These tables can be used in replication configurations where DML statements are sent to replica servers, but the source server does not keep its own copy of the data.

[**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) (also known as [**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)): This clustered database engine is particularly suited for applications that require the highest possible degree of uptime and availability.

[**Merge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-storage-engine): Enables a MySQL DBA or developer to logically group a series of identical **MyISAM** tables and reference them as one object. Good for VLDB environments such as data warehousing.

[**Federated**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-storage-engine): Offers the ability to link separate MySQL servers to create one logical database from many physical servers. Very good for distributed or data mart environments.

[**Example**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#example-storage-engine): This engine serves as an example in the MySQL source code that illustrates how to begin writing new storage engines. It is primarily of interest to developers. The storage engine is a “stub” that does nothing. You can create tables with this engine, but no data can be stored in them or retrieved from them.

You are not restricted to using the same storage engine for an entire server or schema. You can specify the storage engine for any table. For example, an application might use mostly **InnoDB** tables, with one **CSV** table for exporting data to a spreadsheet and a few **MEMORY** tables for temporary workspaces.

***Choosing a Storage Engine***

The various storage engines provided with MySQL are designed with different use cases in mind. The following table provides an overview of some storage engines provided with MySQL, with clarifying notes following the table.

**Table 16.1 Storage Engines Feature Summary**

| **Feature** | **MyISAM** | **Memory** | **InnoDB** | **Archive** | **NDB** |
| --- | --- | --- | --- | --- | --- |
| **B-tree indexes** | Yes | Yes | Yes | No | No |
| **Backup/point-in-time recovery (note 1)** | Yes | Yes | Yes | Yes | Yes |
| **Cluster database support** | No | No | No | No | Yes |
| **Clustered indexes** | No | No | Yes | No | No |
| **Compressed data** | Yes (note 2) | No | Yes | Yes | No |
| **Data caches** | No | N/A | Yes | No | Yes |
| **Encrypted data** | Yes (note 3) | Yes (note 3) | Yes (note 4) | Yes (note 3) | Yes (note 3) |
| **Foreign key support** | No | No | Yes | No | Yes (note 5) |
| **Full-text search indexes** | Yes | No | Yes (note 6) | No | No |
| **Geospatial data type support** | Yes | No | Yes | Yes | Yes |
| **Geospatial indexing support** | Yes | No | Yes (note 7) | No | No |
| **Hash indexes** | No | Yes | No (note 8) | No | Yes |
| **Index caches** | Yes | N/A | Yes | No | Yes |
| **Locking granularity** | Table | Table | Row | Row | Row |
| **MVCC** | No | No | Yes | No | No |
| **Replication support (note 1)** | Yes | Limited (note 9) | Yes | Yes | Yes |
| **Storage limits** | 256TB | RAM | 64TB | None | 384EB |
| **T-tree indexes** | No | No | No | No | Yes |
| **Transactions** | No | No | Yes | No | Yes |
| **Update statistics for data dictionary** | Yes | Yes | Yes | Yes | Yes |

***Notes:***

1. Implemented in the server, rather than in the storage engine.

2. Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.

3. Implemented in the server via encryption functions.

4. Implemented in the server via encryption functions; In MySQL 5.7 and later, data-at-rest encryption is supported.

5. Support for foreign keys is available in MySQL Cluster NDB 7.3 and later.

6. Support for FULLTEXT indexes is available in MySQL 5.6 and later.

7. Support for geospatial indexing is available in MySQL 5.7 and later.

8. InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.

9. See the discussion later in this section.

## 16.1 Setting the Storage Engine

When you create a new table, you can specify which storage engine to use by adding an **ENGINE** table option to the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

-- ENGINE=INNODB not needed unless you have set a different

-- default storage engine.

CREATE TABLE t1 (i INT) ENGINE = INNODB;

-- Simple table definitions can be switched from one to another.

CREATE TABLE t2 (i INT) ENGINE = CSV;

CREATE TABLE t3 (i INT) ENGINE = MEMORY;

When you omit the **ENGINE** option, the default storage engine is used. The default engine is [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) in MySQL 8.0. You can specify the default engine by using the [**--default-storage-engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) server startup option, or by setting the [**default-storage-engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) option in the my.cnf configuration file.

You can set the default storage engine for the current session by setting the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) variable:

SET default\_storage\_engine=NDBCLUSTER;

The storage engine for **TEMPORARY** tables created with [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) can be set separately from the engine for permanent tables by setting the [**default\_tmp\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_tmp_storage_engine), either at startup or at runtime.

To convert a table from one storage engine to another, use an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement that indicates the new engine:

ALTER TABLE t ENGINE = InnoDB;

See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), and [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

If you try to use a storage engine that is not compiled in or that is compiled in but deactivated, MySQL instead creates a table using the default storage engine. For example, in a replication setup, perhaps your source server uses **InnoDB** tables for maximum safety, but the replica servers use other storage engines for speed at the expense of durability or concurrency.

By default, a warning is generated whenever [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) cannot use the default storage engine. To prevent confusing, unintended behavior if the desired engine is unavailable, enable the [**NO\_ENGINE\_SUBSTITUTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_engine_substitution) SQL mode. If the desired engine is unavailable, this setting produces an error instead of a warning, and the table is not created or altered. See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

MySQL may store a table's index and data in one or more other files, depending on the storage engine. Table and column definitions are stored in the MySQL data dictionary. Individual storage engines create any additional files required for the tables that they manage. If a table name contains special characters, the names for the table files contain encoded versions of those characters as described in [Section 9.2.4, “Mapping of Identifiers to File Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifier-mapping).

## 16.2 The MyISAM Storage Engine

[16.2.1 MyISAM Startup Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-start)

[16.2.2 Space Needed for Keys](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#key-space)

[16.2.3 MyISAM Table Storage Formats](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-table-formats)

[16.2.4 MyISAM Table Problems](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-table-problems)

**MyISAM** is based on the older (and no longer available) **ISAM** storage engine but has many useful extensions.

**Table 16.2 MyISAM Storage Engine Features**

| **Feature** | **Support** |
| --- | --- |
| ***B-tree indexes*** | Yes |
| ***Backup/point-in-time recovery*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Cluster database support*** | No |
| ***Clustered indexes*** | No |
| ***Compressed data*** | Yes (Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.) |
| ***Data caches*** | No |
| ***Encrypted data*** | Yes (Implemented in the server via encryption functions.) |
| ***Foreign key support*** | No |
| ***Full-text search indexes*** | Yes |
| ***Geospatial data type support*** | Yes |
| ***Geospatial indexing support*** | Yes |
| ***Hash indexes*** | No |
| ***Index caches*** | Yes |
| ***Locking granularity*** | Table |
| ***MVCC*** | No |
| ***Replication support*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Storage limits*** | 256TB |
| ***T-tree indexes*** | No |
| ***Transactions*** | No |
| ***Update statistics for data dictionary*** | Yes |

Each **MyISAM** table is stored on disk in two files. The files have names that begin with the table name and have an extension to indicate the file type. The data file has an .MYD (**MYData**) extension. The index file has an .MYI (**MYIndex**) extension. The table definition is stored in the MySQL data dictionary.

To specify explicitly that you want a **MyISAM** table, indicate that with an **ENGINE** table option:

CREATE TABLE t (i INT) ENGINE = MYISAM;

In MySQL 8.0, it is normally necessary to use **ENGINE** to specify the **MyISAM** storage engine because **InnoDB** is the default engine.

You can check or repair **MyISAM** tables with the [**mysqlcheck**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlcheck) client or [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) utility. You can also compress **MyISAM** tables with [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack) to take up much less space. See [Section 4.5.3, “mysqlcheck — A Table Maintenance Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlcheck), [Section 4.6.4, “myisamchk — MyISAM Table-Maintenance Utility”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk), and [Section 4.6.6, “myisampack — Generate Compressed, Read-Only MyISAM Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack).

In MySQL 8.0, the **MyISAM** storage engine provides no partitioning support. Partitioned ***MyISAM*** tables created in previous versions of MySQL cannot be used in MySQL 8.0. For more information, see [Section 24.6.2, “Partitioning Limitations Relating to Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-storage-engines). For help with upgrading such tables so that they can be used in MySQL 8.0, see [Section 2.11.4, “Changes in MySQL 8.0”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading-from-previous-series).

**MyISAM** tables have the following characteristics:

All data values are stored with the low byte first. This makes the data machine and operating system independent. The only requirements for binary portability are that the machine uses two's-complement signed integers and IEEE floating-point format. These requirements are widely used among mainstream machines. Binary compatibility might not be applicable to embedded systems, which sometimes have peculiar processors.

There is no significant speed penalty for storing data low byte first; the bytes in a table row normally are unaligned and it takes little more processing to read an unaligned byte in order than in reverse order. Also, the code in the server that fetches column values is not time critical compared to other code.

All numeric key values are stored with the high byte first to permit better index compression.

Large files (up to 63-bit file length) are supported on file systems and operating systems that support large files.

There is a limit of (232)2 (1.844E+19) rows in a **MyISAM** table.

The maximum number of indexes per **MyISAM** table is 64.

The maximum number of columns per index is 16.

The maximum key length is 1000 bytes. This can also be changed by changing the source and recompiling. For the case of a key longer than 250 bytes, a larger key block size than the default of 1024 bytes is used.

When rows are inserted in sorted order (as when you are using an **AUTO\_INCREMENT** column), the index tree is split so that the high node only contains one key. This improves space utilization in the index tree.

Internal handling of one **AUTO\_INCREMENT** column per table is supported. **MyISAM** automatically updates this column for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operations. This makes **AUTO\_INCREMENT** columns faster (at least 10%). Values at the top of the sequence are not reused after being deleted. (When an **AUTO\_INCREMENT** column is defined as the last column of a multiple-column index, reuse of values deleted from the top of a sequence does occur.) The **AUTO\_INCREMENT** value can be reset with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

Dynamic-sized rows are much less fragmented when mixing deletes with updates and inserts. This is done by automatically combining adjacent deleted blocks and by extending blocks if the next block is deleted.

**MyISAM** supports concurrent inserts: If a table has no free blocks in the middle of the data file, you can [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) new rows into it at the same time that other threads are reading from the table. A free block can occur as a result of deleting rows or an update of a dynamic length row with more data than its current contents. When all free blocks are used up (filled in), future inserts become concurrent again. See [Section 8.11.3, “Concurrent Inserts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#concurrent-inserts).

You can put the data file and index file in different directories on different physical devices to get more speed with the **DATA DIRECTORY** and **INDEX DIRECTORY** table options to [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

[**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns can be indexed.

**NULL** values are permitted in indexed columns. This takes 0 to 1 bytes per key.

Each character column can have a different character set. See [Chapter 10, *Character Sets, Collations, Unicode*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html).

There is a flag in the **MyISAM** index file that indicates whether the table was closed correctly. If [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started with the [**myisam\_recover\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_recover_options) system variable set, **MyISAM** tables are automatically checked when opened, and are repaired if the table wasn't closed properly.

[**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) marks tables as checked if you run it with the [--update-state](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_myisamchk_update-state) option. [**myisamchk --fast**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) checks only those tables that don't have this mark.

[**myisamchk --analyze**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) stores statistics for portions of keys, as well as for entire keys.

[**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack) can pack [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns.

**MyISAM** also supports the following features:

Support for a true [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) type; a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column starts with a length stored in one or two bytes.

Tables with [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns may have fixed or dynamic row length.

The sum of the lengths of the [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) and [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns in a table may be up to 64KB.

Arbitrary length **UNIQUE** constraints.

### Additional Resources

A forum dedicated to the **MyISAM** storage engine is available at <https://forums.mysql.com/list.php?21>.

### 16.2.1 MyISAM Startup Options

The following options to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) can be used to change the behavior of **MyISAM** tables. For additional information, see [Section 5.1.7, “Server Command Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-options).

**Table 16.3 MyISAM Option and Variable Reference**

| **Name** | **Cmd-Line** | **Option File** | **System Var** | **Status Var** | **Var Scope** | **Dynamic** |
| --- | --- | --- | --- | --- | --- | --- |
| [**bulk\_insert\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_bulk_insert_buffer_size) | Yes | Yes | Yes |  | Both | Yes |
| [**concurrent\_insert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_concurrent_insert) | Yes | Yes | Yes |  | Global | Yes |
| [**delay\_key\_write**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_delay_key_write) | Yes | Yes | Yes |  | Global | Yes |
| [**have\_rtree\_keys**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_have_rtree_keys) |  |  | Yes |  | Global | No |
| [**key\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_key_buffer_size) | Yes | Yes | Yes |  | Global | Yes |
| [**log-isam**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_log-isam) | Yes | Yes |  |  |  |  |
| [**myisam-block-size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_myisam-block-size) | Yes | Yes |  |  |  |  |
| [**myisam\_data\_pointer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_data_pointer_size) | Yes | Yes | Yes |  | Global | Yes |
| [**myisam\_max\_sort\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_max_sort_file_size) | Yes | Yes | Yes |  | Global | Yes |
| [**myisam\_mmap\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_mmap_size) | Yes | Yes | Yes |  | Global | No |
| [**myisam\_recover\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_recover_options) | Yes | Yes | Yes |  | Global | No |
| [**myisam\_repair\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_repair_threads) | Yes | Yes | Yes |  | Both | Yes |
| [**myisam\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_sort_buffer_size) | Yes | Yes | Yes |  | Both | Yes |
| [**myisam\_stats\_method**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_stats_method) | Yes | Yes | Yes |  | Both | Yes |
| [**myisam\_use\_mmap**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_use_mmap) | Yes | Yes | Yes |  | Global | Yes |
| [**tmp\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmp_table_size) | Yes | Yes | Yes |  | Both | Yes |

The following system variables affect the behavior of **MyISAM** tables. For additional information, see [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

[**bulk\_insert\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_bulk_insert_buffer_size)

The size of the tree cache used in bulk insert optimization.

**Note**

This is a limit per thread!

[delay\_key\_write=ALL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_delay_key_write)

Don't flush key buffers between writes for any **MyISAM** table.

**Note**

If you do this, you should not access **MyISAM** tables from another program (such as from another MySQL server or with [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk)) when the tables are in use. Doing so risks index corruption. Using [--external-locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_external-locking) does not eliminate this risk.

[**myisam\_max\_sort\_file\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_max_sort_file_size)

The maximum size of the temporary file that MySQL is permitted to use while re-creating a **MyISAM** index (during [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data)). If the file size would be larger than this value, the index is created using the key cache instead, which is slower. The value is given in bytes.

[**myisam\_recover\_options=*mode***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_recover_options)

Set the mode for automatic recovery of crashed **MyISAM** tables.

[**myisam\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_sort_buffer_size)

Set the size of the buffer used when recovering tables.

Automatic recovery is activated if you start [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with the [**myisam\_recover\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_recover_options) system variable set. In this case, when the server opens a **MyISAM** table, it checks whether the table is marked as crashed or whether the open count variable for the table is not 0 and you are running the server with external locking disabled. If either of these conditions is true, the following happens:

The server checks the table for errors.

If the server finds an error, it tries to do a fast table repair (with sorting and without re-creating the data file).

If the repair fails because of an error in the data file (for example, a duplicate-key error), the server tries again, this time re-creating the data file.

If the repair still fails, the server tries once more with the old repair option method (write row by row without sorting). This method should be able to repair any type of error and has low disk space requirements.

If the recovery wouldn't be able to recover all rows from previously completed statements and you didn't specify **FORCE** in the value of the [**myisam\_recover\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_recover_options) system variable, automatic repair aborts with an error message in the error log:

Error: Couldn't repair table: test.g00pages

If you specify **FORCE**, a warning like this is written instead:

Warning: Found 344 of 354 rows when repairing ./test/g00pages

If the automatic recovery value includes **BACKUP**, the recovery process creates files with names of the form ***tbl\_name-datetime***.BAK. You should have a **cron** script that automatically moves these files from the database directories to backup media.

### 16.2.2 Space Needed for Keys

**MyISAM** tables use B-tree indexes. You can roughly calculate the size for the index file as **(key\_length+4)/0.67**, summed over all keys. This is for the worst case when all keys are inserted in sorted order and the table doesn't have any compressed keys.

String indexes are space compressed. If the first index part is a string, it is also prefix compressed. Space compression makes the index file smaller than the worst-case figure if a string column has a lot of trailing space or is a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column that is not always used to the full length. Prefix compression is used on keys that start with a string. Prefix compression helps if there are many strings with an identical prefix.

In **MyISAM** tables, you can also prefix compress numbers by specifying the **PACK\_KEYS=1** table option when you create the table. Numbers are stored with the high byte first, so this helps when you have many integer keys that have an identical prefix.

### 16.2.3 MyISAM Table Storage Formats

[16.2.3.1 Static (Fixed-Length) Table Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#static-format)

[16.2.3.2 Dynamic Table Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#dynamic-format)

[16.2.3.3 Compressed Table Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#compressed-format)

**MyISAM** supports three different storage formats. Two of them, fixed and dynamic format, are chosen automatically depending on the type of columns you are using. The third, compressed format, can be created only with the [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack) utility (see [Section 4.6.6, “myisampack — Generate Compressed, Read-Only MyISAM Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack)).

When you use [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) for a table that has no [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, you can force the table format to **FIXED** or **DYNAMIC** with the **ROW\_FORMAT** table option.

See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), for information about **ROW\_FORMAT**.

You can decompress (unpack) compressed **MyISAM** tables using [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) [--unpack](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_myisamchk_unpack); see [Section 4.6.4, “myisamchk — MyISAM Table-Maintenance Utility”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk), for more information.

#### 16.2.3.1 Static (Fixed-Length) Table Characteristics

Static format is the default for **MyISAM** tables. It is used when the table contains no variable-length columns ([**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). Each row is stored using a fixed number of bytes.

Of the three **MyISAM** storage formats, static format is the simplest and most secure (least subject to corruption). It is also the fastest of the on-disk formats due to the ease with which rows in the data file can be found on disk: To look up a row based on a row number in the index, multiply the row number by the row length to calculate the row position. Also, when scanning a table, it is very easy to read a constant number of rows with each disk read operation.

The security is evidenced if your computer crashes while the MySQL server is writing to a fixed-format **MyISAM** file. In this case, [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) can easily determine where each row starts and ends, so it can usually reclaim all rows except the partially written one. **MyISAM** table indexes can always be reconstructed based on the data rows.

**Note**

Fixed-length row format is available only for tables having no [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. Creating a table having such columns with an explicit **ROW\_FORMAT** clause does not raise an error or warning; the format specification is ignored.

Static-format tables have these characteristics:

[**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) and [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns are space-padded to the specified column width, although the column type is not altered. [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) and [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) columns are padded with **0x00** bytes to the column width.

**NULL** columns require additional space in the row to record whether their values are **NULL**. Each **NULL** column takes one bit extra, rounded up to the nearest byte.

Very quick.

Easy to cache.

Easy to reconstruct after a crash, because rows are located in fixed positions.

Reorganization is unnecessary unless you delete a huge number of rows and want to return free disk space to the operating system. To do this, use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) or [**myisamchk -r**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

Usually require more disk space than dynamic-format tables.

The expected row length in bytes for static-sized rows is calculated using the following expression:

row length = 1

+ (***sum of column lengths***)

+ (***number of NULL columns*** + ***delete\_flag*** + 7)/8

+ (***number of variable-length columns***)

***delete\_flag*** is 1 for tables with static row format. Static tables use a bit in the row record for a flag that indicates whether the row has been deleted. ***delete\_flag*** is 0 for dynamic tables because the flag is stored in the dynamic row header.

#### 16.2.3.2 Dynamic Table Characteristics

Dynamic storage format is used if a **MyISAM** table contains any variable-length columns ([**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)), or if the table was created with the **ROW\_FORMAT=DYNAMIC** table option.

Dynamic format is a little more complex than static format because each row has a header that indicates how long it is. A row can become fragmented (stored in noncontiguous pieces) when it is made longer as a result of an update.

You can use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) or [**myisamchk -r**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) to defragment a table. If you have fixed-length columns that you access or change frequently in a table that also contains some variable-length columns, it might be a good idea to move the variable-length columns to other tables just to avoid fragmentation.

Dynamic-format tables have these characteristics:

All string columns are dynamic except those with a length less than four.

Each row is preceded by a bitmap that indicates which columns contain the empty string (for string columns) or zero (for numeric columns). This does not include columns that contain **NULL** values. If a string column has a length of zero after trailing space removal, or a numeric column has a value of zero, it is marked in the bitmap and not saved to disk. Nonempty strings are saved as a length byte plus the string contents.

**NULL** columns require additional space in the row to record whether their values are **NULL**. Each **NULL** column takes one bit extra, rounded up to the nearest byte.

Much less disk space usually is required than for fixed-length tables.

Each row uses only as much space as is required. However, if a row becomes larger, it is split into as many pieces as are required, resulting in row fragmentation. For example, if you update a row with information that extends the row length, the row becomes fragmented. In this case, you may have to run [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) or [**myisamchk -r**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) from time to time to improve performance. Use [**myisamchk -ei**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) to obtain table statistics.

More difficult than static-format tables to reconstruct after a crash, because rows may be fragmented into many pieces and links (fragments) may be missing.

The expected row length for dynamic-sized rows is calculated using the following expression:

3

+ (***number of columns*** + 7) / 8

+ (***number of char columns***)

+ (***packed size of numeric columns***)

+ (***length of strings***)

+ (***number of NULL columns*** + 7) / 8

There is a penalty of 6 bytes for each link. A dynamic row is linked whenever an update causes an enlargement of the row. Each new link is at least 20 bytes, so the next enlargement probably goes in the same link. If not, another link is created. You can find the number of links using [**myisamchk -ed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk). All links may be removed with [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) or [**myisamchk -r**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

#### 16.2.3.3 Compressed Table Characteristics

Compressed storage format is a read-only format that is generated with the [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack) tool. Compressed tables can be uncompressed with [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

Compressed tables have the following characteristics:

Compressed tables take very little disk space. This minimizes disk usage, which is helpful when using slow disks (such as CD-ROMs).

Each row is compressed separately, so there is very little access overhead. The header for a row takes up one to three bytes depending on the biggest row in the table. Each column is compressed differently. There is usually a different Huffman tree for each column. Some of the compression types are:

Suffix space compression.

Prefix space compression.

Numbers with a value of zero are stored using one bit.

If values in an integer column have a small range, the column is stored using the smallest possible type. For example, a [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) column (eight bytes) can be stored as a [**TINYINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) column (one byte) if all its values are in the range from **-128** to **127**.

If a column has only a small set of possible values, the data type is converted to [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum).

A column may use any combination of the preceding compression types.

Can be used for fixed-length or dynamic-length rows.

**Note**

While a compressed table is read only, and you cannot therefore update or add rows in the table, DDL (Data Definition Language) operations are still valid. For example, you may still use **DROP** to drop the table, and [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) to empty the table.

### 16.2.4 MyISAM Table Problems

[16.2.4.1 Corrupted MyISAM Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#corrupted-myisam-tables)

[16.2.4.2 Problems from Tables Not Being Closed Properly](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-table-close)

The file format that MySQL uses to store data has been extensively tested, but there are always circumstances that may cause database tables to become corrupted. The following discussion describes how this can happen and how to handle it.

#### 16.2.4.1 Corrupted MyISAM Tables

Even though the **MyISAM** table format is very reliable (all changes to a table made by an SQL statement are written before the statement returns), you can still get corrupted tables if any of the following events occur:

The [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process is killed in the middle of a write.

An unexpected computer shutdown occurs (for example, the computer is turned off).

Hardware failures.

You are using an external program (such as [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk)) to modify a table that is being modified by the server at the same time.

A software bug in the MySQL or **MyISAM** code.

Typical symptoms of a corrupt table are:

You get the following error while selecting data from the table:

Incorrect key file for table: '...'. Try to repair it

Queries don't find rows in the table or return incomplete results.

You can check the health of a **MyISAM** table using the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) statement, and repair a corrupted **MyISAM** table with [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table). When [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is not running, you can also check or repair a table with the [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) command. See [Section 13.7.3.2, “CHECK TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table), [Section 13.7.3.5, “REPAIR TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), and [Section 4.6.4, “myisamchk — MyISAM Table-Maintenance Utility”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

If your tables become corrupted frequently, you should try to determine why this is happening. The most important thing to know is whether the table became corrupted as a result of an unexpected server exit. You can verify this easily by looking for a recent **restarted mysqld** message in the error log. If there is such a message, it is likely that table corruption is a result of the server dying. Otherwise, corruption may have occurred during normal operation. This is a bug. You should try to create a reproducible test case that demonstrates the problem. See [Section B.3.3.3, “What to Do If MySQL Keeps Crashing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#crashing), and [Section 5.9, “Debugging MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#debugging-mysql).

#### 16.2.4.2 Problems from Tables Not Being Closed Properly

Each **MyISAM** index file (.MYI file) has a counter in the header that can be used to check whether a table has been closed properly. If you get the following warning from [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) or [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk), it means that this counter has gone out of sync:

clients are using or haven't closed the table properly

This warning doesn't necessarily mean that the table is corrupted, but you should at least check the table.

The counter works as follows:

The first time a table is updated in MySQL, a counter in the header of the index files is incremented.

The counter is not changed during further updates.

When the last instance of a table is closed (because a [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) operation was performed or because there is no room in the table cache), the counter is decremented if the table has been updated at any point.

When you repair the table or check the table and it is found to be okay, the counter is reset to zero.

To avoid problems with interaction with other processes that might check the table, the counter is not decremented on close if it was zero.

In other words, the counter can become incorrect only under these conditions:

A **MyISAM** table is copied without first issuing [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) and [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables).

MySQL has crashed between an update and the final close. (The table may still be okay because MySQL always issues writes for everything between each statement.)

A table was modified by [**myisamchk --recover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) or [**myisamchk --update-state**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) at the same time that it was in use by [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld).

Multiple [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) servers are using the table and one server performed a [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) or [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) on the table while it was in use by another server. In this setup, it is safe to use [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table), although you might get the warning from other servers. However, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) should be avoided because when one server replaces the data file with a new one, this is not known to the other servers.

In general, it is a bad idea to share a data directory among multiple servers. See [Section 5.8, “Running Multiple MySQL Instances on One Machine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#multiple-servers), for additional discussion.

## 16.3 The MEMORY Storage Engine

The **MEMORY** storage engine (formerly known as **HEAP**) creates special-purpose tables with contents that are stored in memory. Because the data is vulnerable to crashes, hardware issues, or power outages, only use these tables as temporary work areas or read-only caches for data pulled from other tables.

**Table 16.4 MEMORY Storage Engine Features**

| **Feature** | **Support** |
| --- | --- |
| ***B-tree indexes*** | Yes |
| ***Backup/point-in-time recovery*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Cluster database support*** | No |
| ***Clustered indexes*** | No |
| ***Compressed data*** | No |
| ***Data caches*** | N/A |
| ***Encrypted data*** | Yes (Implemented in the server via encryption functions.) |
| ***Foreign key support*** | No |
| ***Full-text search indexes*** | No |
| ***Geospatial data type support*** | No |
| ***Geospatial indexing support*** | No |
| ***Hash indexes*** | Yes |
| ***Index caches*** | N/A |
| ***Locking granularity*** | Table |
| ***MVCC*** | No |
| ***Replication support*** (Implemented in the server, rather than in the storage engine.) | Limited (See the discussion later in this section.) |
| ***Storage limits*** | RAM |
| ***T-tree indexes*** | No |
| ***Transactions*** | No |
| ***Update statistics for data dictionary*** | Yes |

[When to Use MEMORY or NDB Cluster](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-compared-cluster)

[Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-partitioning)

[Performance Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-performance-characteristics)

[Characteristics of MEMORY Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-characteristics-of-memory-tables)

[DDL Operations for MEMORY Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-ddl-operations-for-memory-tables)

[Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-indexes)

[User-Created and Temporary Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-user-created-and-temporary-tables)

[Loading Data](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-loading-data)

[MEMORY Tables and Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-tables-replication)

[Managing Memory Use](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-managing-memory-use)

[Additional Resources](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine-additional-resources)

### When to Use MEMORY or NDB Cluster

Developers looking to deploy applications that use the **MEMORY** storage engine for important, highly available, or frequently updated data should consider whether NDB Cluster is a better choice. A typical use case for the **MEMORY** engine involves these characteristics:

Operations involving transient, non-critical data such as session management or caching. When the MySQL server halts or restarts, the data in **MEMORY** tables is lost.

In-memory storage for fast access and low latency. Data volume can fit entirely in memory without causing the operating system to swap out virtual memory pages.

A read-only or read-mostly data access pattern (limited updates).

NDB Cluster offers the same features as the **MEMORY** engine with higher performance levels, and provides additional features not available with **MEMORY**:

Row-level locking and multiple-thread operation for low contention between clients.

Scalability even with statement mixes that include writes.

Optional disk-backed operation for data durability.

Shared-nothing architecture and multiple-host operation with no single point of failure, enabling 99.999% availability.

Automatic data distribution across nodes; application developers need not craft custom sharding or partitioning solutions.

Support for variable-length data types (including [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) not supported by **MEMORY**.

### Partitioning

**MEMORY** tables cannot be partitioned.

### Performance Characteristics

**MEMORY** performance is constrained by contention resulting from single-thread execution and table lock overhead when processing updates. This limits scalability when load increases, particularly for statement mixes that include writes.

Despite the in-memory processing for **MEMORY** tables, they are not necessarily faster than [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables on a busy server, for general-purpose queries, or under a read/write workload. In particular, the table locking involved with performing updates can slow down concurrent usage of **MEMORY** tables from multiple sessions.

Depending on the kinds of queries performed on a **MEMORY** table, you might create indexes as either the default hash data structure (for looking up single values based on a unique key), or a general-purpose B-tree data structure (for all kinds of queries involving equality, inequality, or range operators such as less than or greater than). The following sections illustrate the syntax for creating both kinds of indexes. A common performance issue is using the default hash indexes in workloads where B-tree indexes are more efficient.

### Characteristics of MEMORY Tables

The **MEMORY** storage engine does not create any files on disk. The table definition is stored in the MySQL data dictionary.

**MEMORY** tables have the following characteristics:

Space for **MEMORY** tables is allocated in small blocks. Tables use 100% dynamic hashing for inserts. No overflow area or extra key space is needed. No extra space is needed for free lists. Deleted rows are put in a linked list and are reused when you insert new data into the table. **MEMORY** tables also have none of the problems commonly associated with deletes plus inserts in hashed tables.

**MEMORY** tables use a fixed-length row-storage format. Variable-length types such as [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) are stored using a fixed length.

**MEMORY** tables cannot contain [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns.

**MEMORY** includes support for **AUTO\_INCREMENT** columns.

Non-**TEMPORARY** **MEMORY** tables are shared among all clients, just like any other non-**TEMPORARY** table.

### DDL Operations for MEMORY Tables

To create a **MEMORY** table, specify the clause **ENGINE=MEMORY** on the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

CREATE TABLE t (i INT) ENGINE = MEMORY;

As indicated by the engine name, **MEMORY** tables are stored in memory. They use hash indexes by default, which makes them very fast for single-value lookups, and very useful for creating temporary tables. However, when the server shuts down, all rows stored in **MEMORY** tables are lost. The tables themselves continue to exist because their definitions are stored in the MySQL data dictionary, but they are empty when the server restarts.

This example shows how you might create, use, and remove a **MEMORY** table:

mysql> **CREATE TABLE test ENGINE=MEMORY**

**SELECT ip,SUM(downloads) AS down**

**FROM log\_table GROUP BY ip;**

mysql> **SELECT COUNT(ip),AVG(down) FROM test;**

mysql> **DROP TABLE test;**

The maximum size of **MEMORY** tables is limited by the [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) system variable, which has a default value of 16MB. To enforce different size limits for **MEMORY** tables, change the value of this variable. The value in effect for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), or a subsequent [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), is the value used for the life of the table. A server restart also sets the maximum size of existing **MEMORY** tables to the global [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) value. You can set the size for individual tables as described later in this section.

### Indexes

The **MEMORY** storage engine supports both **HASH** and **BTREE** indexes. You can specify one or the other for a given index by adding a **USING** clause as shown here:

CREATE TABLE lookup

(id INT, INDEX USING HASH (id))

ENGINE = MEMORY;

CREATE TABLE lookup

(id INT, INDEX USING BTREE (id))

ENGINE = MEMORY;

For general characteristics of B-tree and hash indexes, see [Section 8.3.1, “How MySQL Uses Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#mysql-indexes).

**MEMORY** tables can have up to 64 indexes per table, 16 columns per index and a maximum key length of 3072 bytes.

If a **MEMORY** table hash index has a high degree of key duplication (many index entries containing the same value), updates to the table that affect key values and all deletes are significantly slower. The degree of this slowdown is proportional to the degree of duplication (or, inversely proportional to the index cardinality). You can use a **BTREE** index to avoid this problem.

**MEMORY** tables can have nonunique keys. (This is an uncommon feature for implementations of hash indexes.)

Columns that are indexed can contain **NULL** values.

### User-Created and Temporary Tables

**MEMORY** table contents are stored in memory, which is a property that **MEMORY** tables share with internal temporary tables that the server creates on the fly while processing queries. However, the two types of tables differ in that **MEMORY** tables are not subject to storage conversion, whereas internal temporary tables are:

If an internal temporary table becomes too large, the server automatically converts it to on-disk storage, as described in [Section 8.4.4, “Internal Temporary Table Use in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-temporary-tables).

User-created **MEMORY** tables are never converted to disk tables.

### Loading Data

To populate a **MEMORY** table when the MySQL server starts, you can use the [**init\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_file) system variable. For example, you can put statements such as [**INSERT INTO ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) or [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) into a file to load the table from a persistent data source, and use [**init\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_file) to name the file. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables), and [Section 13.2.7, “LOAD DATA Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data).

### MEMORY Tables and Replication

When a replication source server shuts down and restarts, its [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables become empty. To replicate this effect to replicas, the first time that the source uses a given [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) table after startup, it logs an event that notifies replicas that the table must be emptied by writing a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or (from MySQL 8.0.22) [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement for that table to the binary log. When a replica server shuts down and restarts, its [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables also become empty, and it writes a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or (from MySQL 8.0.22) [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement to its own binary log, which is passed on to any downstream replicas.

When you use [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables in a replication topology, in some situations, the table on the source and the table on the replica can differ. For information on handling each of these situations to prevent stale reads or errors, see [Section 17.5.1.21, “Replication and MEMORY Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-memory).

### Managing Memory Use

The server needs sufficient memory to maintain all **MEMORY** tables that are in use at the same time.

Memory is not reclaimed if you delete individual rows from a **MEMORY** table. Memory is reclaimed only when the entire table is deleted. Memory that was previously used for deleted rows is re-used for new rows within the same table. To free all the memory used by a **MEMORY** table when you no longer require its contents, execute [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) to remove all rows, or remove the table altogether using [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table). To free up the memory used by deleted rows, use **ALTER TABLE ENGINE=MEMORY** to force a table rebuild.

The memory needed for one row in a **MEMORY** table is calculated using the following expression:

SUM\_OVER\_ALL\_BTREE\_KEYS(***max\_length\_of\_key*** + sizeof(char\*) \* 4)

+ SUM\_OVER\_ALL\_HASH\_KEYS(sizeof(char\*) \* 2)

+ ALIGN(***length\_of\_row***+1, sizeof(char\*))

**ALIGN()** represents a round-up factor to cause the row length to be an exact multiple of the **char** pointer size. **sizeof(char\*)** is 4 on 32-bit machines and 8 on 64-bit machines.

As mentioned earlier, the [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) system variable sets the limit on the maximum size of **MEMORY** tables. To control the maximum size for individual tables, set the session value of this variable before creating each table. (Do not change the global [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) value unless you intend the value to be used for **MEMORY** tables created by all clients.) The following example creates two **MEMORY** tables, with a maximum size of 1MB and 2MB, respectively:

mysql> **SET max\_heap\_table\_size = 1024\*1024;**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t1 (id INT, UNIQUE(id)) ENGINE = MEMORY;**

Query OK, 0 rows affected (0.01 sec)

mysql> **SET max\_heap\_table\_size = 1024\*1024\*2;**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE TABLE t2 (id INT, UNIQUE(id)) ENGINE = MEMORY;**

Query OK, 0 rows affected (0.00 sec)

Both tables revert to the server's global [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) value if the server restarts.

You can also specify a **MAX\_ROWS** table option in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements for **MEMORY** tables to provide a hint about the number of rows you plan to store in them. This does not enable the table to grow beyond the [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) value, which still acts as a constraint on maximum table size. For maximum flexibility in being able to use **MAX\_ROWS**, set [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) at least as high as the value to which you want each **MEMORY** table to be able to grow.

### Additional Resources

A forum dedicated to the **MEMORY** storage engine is available at <https://forums.mysql.com/list.php?92>.

## 16.4 The CSV Storage Engine

[16.4.1 Repairing and Checking CSV Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#se-csv-repair)

[16.4.2 CSV Limitations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#se-csv-limitations)

The **CSV** storage engine stores data in text files using comma-separated values format.

The **CSV** storage engine is always compiled into the MySQL server.

To examine the source for the **CSV** engine, look in the storage/csv directory of a MySQL source distribution.

When you create a **CSV** table, the server creates a plain text data file having a name that begins with the table name and has a .CSV extension. When you store data into the table, the storage engine saves it into the data file in comma-separated values format.

mysql> **CREATE TABLE test (i INT NOT NULL, c CHAR(10) NOT NULL)**

**ENGINE = CSV;**

Query OK, 0 rows affected (0.06 sec)

mysql> **INSERT INTO test VALUES(1,'record one'),(2,'record two');**

Query OK, 2 rows affected (0.05 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM test;**

+---+------------+

| i | c |

+---+------------+

| 1 | record one |

| 2 | record two |

+---+------------+

2 rows in set (0.00 sec)

Creating a **CSV** table also creates a corresponding metafile that stores the state of the table and the number of rows that exist in the table. The name of this file is the same as the name of the table with the extension CSM.

If you examine the test.CSV file in the database directory created by executing the preceding statements, its contents should look like this:

"1","record one"

"2","record two"

This format can be read, and even written, by spreadsheet applications such as Microsoft Excel.

### 16.4.1 Repairing and Checking CSV Tables

The **CSV** storage engine supports the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) and [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statements to verify and, if possible, repair a damaged **CSV** table.

When running the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) statement, the **CSV** file is checked for validity by looking for the correct field separators, escaped fields (matching or missing quotation marks), the correct number of fields compared to the table definition and the existence of a corresponding **CSV** metafile. The first invalid row discovered causes an error. Checking a valid table produces output like that shown here:

mysql> **CHECK TABLE csvtest;**

+--------------+-------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+--------------+-------+----------+----------+

| test.csvtest | check | status | OK |

+--------------+-------+----------+----------+

A check on a corrupted table returns a fault such as

mysql> **CHECK TABLE csvtest;**

+--------------+-------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+--------------+-------+----------+----------+

| test.csvtest | check | error | Corrupt |

+--------------+-------+----------+----------+

To repair a table, use [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), which copies as many valid rows from the existing **CSV** data as possible, and then replaces the existing **CSV** file with the recovered rows. Any rows beyond the corrupted data are lost.

mysql> **REPAIR TABLE csvtest;**

+--------------+--------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+--------------+--------+----------+----------+

| test.csvtest | repair | status | OK |

+--------------+--------+----------+----------+

**Warning**

During repair, only the rows from the **CSV** file up to the first damaged row are copied to the new table. All other rows from the first damaged row to the end of the table are removed, even valid rows.

### 16.4.2 CSV Limitations

The **CSV** storage engine does not support indexing.

The **CSV** storage engine does not support partitioning.

All tables that you create using the **CSV** storage engine must have the **NOT NULL** attribute on all columns.

## 16.5 The ARCHIVE Storage Engine

The **ARCHIVE** storage engine produces special-purpose tables that store large amounts of unindexed data in a very small footprint.

**Table 16.5 ARCHIVE Storage Engine Features**

| **Feature** | **Support** |
| --- | --- |
| ***B-tree indexes*** | No |
| ***Backup/point-in-time recovery*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Cluster database support*** | No |
| ***Clustered indexes*** | No |
| ***Compressed data*** | Yes |
| ***Data caches*** | No |
| ***Encrypted data*** | Yes (Implemented in the server via encryption functions.) |
| ***Foreign key support*** | No |
| ***Full-text search indexes*** | No |
| ***Geospatial data type support*** | Yes |
| ***Geospatial indexing support*** | No |
| ***Hash indexes*** | No |
| ***Index caches*** | No |
| ***Locking granularity*** | Row |
| ***MVCC*** | No |
| ***Replication support*** (Implemented in the server, rather than in the storage engine.) | Yes |
| ***Storage limits*** | None |
| ***T-tree indexes*** | No |
| ***Transactions*** | No |
| ***Update statistics for data dictionary*** | Yes |

The **ARCHIVE** storage engine is included in MySQL binary distributions. To enable this storage engine if you build MySQL from source, invoke **CMake** with the [-DWITH\_ARCHIVE\_STORAGE\_ENGINE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_storage_engine_options) option.

To examine the source for the **ARCHIVE** engine, look in the storage/archive directory of a MySQL source distribution.

You can check whether the **ARCHIVE** storage engine is available with the [**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines) statement.

When you create an **ARCHIVE** table, the storage engine creates files with names that begin with the table name. The data file has an extension of .ARZ. An .ARN file may appear during optimization operations.

The **ARCHIVE** engine supports [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), but not [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). It does support **ORDER BY** operations, [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, and spatial data types (see [Section 11.4.1, “Spatial Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-type-overview)). Geographic spatial reference systems are not supported. The **ARCHIVE** engine uses row-level locking.

The **ARCHIVE** engine supports the **AUTO\_INCREMENT** column attribute. The **AUTO\_INCREMENT** column can have either a unique or nonunique index. Attempting to create an index on any other column results in an error. The **ARCHIVE** engine also supports the **AUTO\_INCREMENT** table option in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements to specify the initial sequence value for a new table or reset the sequence value for an existing table, respectively.

**ARCHIVE** does not support inserting a value into an **AUTO\_INCREMENT** column less than the current maximum column value. Attempts to do so result in an [**ER\_DUP\_KEY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_dup_key) error.

The **ARCHIVE** engine ignores [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns if they are not requested and scans past them while reading.

The **ARCHIVE** storage engine does not support partitioning.

***Storage:*** Rows are compressed as they are inserted. The **ARCHIVE** engine uses **zlib** lossless data compression (see <http://www.zlib.net/>). You can use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) to analyze the table and pack it into a smaller format (for a reason to use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), see later in this section). The engine also supports [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table). There are several types of insertions that are used:

An [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement just pushes rows into a compression buffer, and that buffer flushes as necessary. The insertion into the buffer is protected by a lock. A [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) forces a flush to occur.

A bulk insert is visible only after it completes, unless other inserts occur at the same time, in which case it can be seen partially. A [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) never causes a flush of a bulk insert unless a normal insert occurs while it is loading.

***Retrieval***: On retrieval, rows are uncompressed on demand; there is no row cache. A [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) operation performs a complete table scan: When a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) occurs, it finds out how many rows are currently available and reads that number of rows. [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is performed as a consistent read. Note that lots of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements during insertion can deteriorate the compression, unless only bulk inserts are used. To achieve better compression, you can use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) or [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table). The number of rows in **ARCHIVE** tables reported by [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) is always accurate. See [Section 13.7.3.4, “OPTIMIZE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), [Section 13.7.3.5, “REPAIR TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), and [Section 13.7.7.38, “SHOW TABLE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status).

### Additional Resources

A forum dedicated to the **ARCHIVE** storage engine is available at <https://forums.mysql.com/list.php?112>.

## 16.6 The BLACKHOLE Storage Engine

The **BLACKHOLE** storage engine acts as a “black hole” that accepts data but throws it away and does not store it. Retrievals always return an empty result:

mysql> **CREATE TABLE test(i INT, c CHAR(10)) ENGINE = BLACKHOLE;**

Query OK, 0 rows affected (0.03 sec)

mysql> **INSERT INTO test VALUES(1,'record one'),(2,'record two');**

Query OK, 2 rows affected (0.00 sec)

Records: 2 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM test;**

Empty set (0.00 sec)

To enable the **BLACKHOLE** storage engine if you build MySQL from source, invoke **CMake** with the [-DWITH\_BLACKHOLE\_STORAGE\_ENGINE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_storage_engine_options) option.

To examine the source for the **BLACKHOLE** engine, look in the sql directory of a MySQL source distribution.

When you create a **BLACKHOLE** table, the server creates the table definition in the global data dictionary. There are no files associated with the table.

The **BLACKHOLE** storage engine supports all kinds of indexes. That is, you can include index declarations in the table definition.

The maximum key length is 3072 bytes as of MySQL 8.0.26. Prior to 8.0.26, the maximum key length is 1000 bytes.

The **BLACKHOLE** storage engine does not support partitioning.

You can check whether the **BLACKHOLE** storage engine is available with the [**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines) statement.

Inserts into a **BLACKHOLE** table do not store any data, but if statement based binary logging is enabled, the SQL statements are logged and replicated to replica servers. This can be useful as a repeater or filter mechanism.

Suppose that your application requires replica-side filtering rules, but transferring all binary log data to the replica first results in too much traffic. In such a case, it is possible to set up on the replication source server a “dummy” replica process whose default storage engine is **BLACKHOLE**, depicted as follows:

**Figure 16.1 Replication using BLACKHOLE for Filtering**

The source writes to its binary log. The “dummy” [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process acts as a replica, applying the desired combination of **replicate-do-\*** and **replicate-ignore-\*** rules, and writes a new, filtered binary log of its own. (See [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options).) This filtered log is provided to the replica.

The dummy process does not actually store any data, so there is little processing overhead incurred by running the additional [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process on the replication source server. This type of setup can be repeated with additional replicas.

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) triggers for **BLACKHOLE** tables work as expected. However, because the **BLACKHOLE** table does not actually store any data, [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) triggers are not activated: The **FOR EACH ROW** clause in the trigger definition does not apply because there are no rows.

Other possible uses for the **BLACKHOLE** storage engine include:

Verification of dump file syntax.

Measurement of the overhead from binary logging, by comparing performance using **BLACKHOLE** with and without binary logging enabled.

**BLACKHOLE** is essentially a “no-op” storage engine, so it could be used for finding performance bottlenecks not related to the storage engine itself.

The **BLACKHOLE** engine is transaction-aware, in the sense that committed transactions are written to the binary log and rolled-back transactions are not.

***Blackhole Engine and Auto Increment Columns***

The **BLACKHOLE** engine is a no-op engine. Any operations performed on a table using **BLACKHOLE** have no effect. This should be borne in mind when considering the behavior of primary key columns that auto increment. The engine does not automatically increment field values, and does not retain auto increment field state. This has important implications in replication.

Consider the following replication scenario where all three of the following conditions apply:

On a source server there is a blackhole table with an auto increment field that is a primary key.

On a replica the same table exists but using the MyISAM engine.

Inserts are performed into the source's table without explicitly setting the auto increment value in the **INSERT** statement itself or through using a **SET INSERT\_ID** statement.

In this scenario replication fails with a duplicate entry error on the primary key column.

In statement based replication, the value of **INSERT\_ID** in the context event is always the same. Replication therefore fails due to trying insert a row with a duplicate value for a primary key column.

In row based replication, the value that the engine returns for the row always be the same for each insert. This results in the replica attempting to replay two insert log entries using the same value for the primary key column, and so replication fails.

***Column Filtering***

When using row-based replication, ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), a replica where the last columns are missing from a table is supported, as described in the section [Section 17.5.1.9, “Replication with Differing Table Definitions on Source and Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-differing-tables).

This filtering works on the replica side, that is, the columns are copied to the replica before they are filtered out. There are at least two cases where it is not desirable to copy the columns to the replica:

If the data is confidential, so the replica server should not have access to it.

If the source has many replicas, filtering before sending to the replicas may reduce network traffic.

Source column filtering can be achieved using the **BLACKHOLE** engine. This is carried out in a way similar to how source table filtering is achieved - by using the **BLACKHOLE** engine and the [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) or [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table) option.

The setup for the source is:

CREATE TABLE t1 (public\_col\_1, ..., public\_col\_N,

secret\_col\_1, ..., secret\_col\_M) ENGINE=MyISAM;

The setup for the trusted replica is:

CREATE TABLE t1 (public\_col\_1, ..., public\_col\_N) ENGINE=BLACKHOLE;

The setup for the untrusted replica is:

CREATE TABLE t1 (public\_col\_1, ..., public\_col\_N) ENGINE=MyISAM;

## 16.7 The MERGE Storage Engine

[16.7.1 MERGE Table Advantages and Disadvantages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-table-advantages)

[16.7.2 MERGE Table Problems](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-table-problems)

The **MERGE** storage engine, also known as the **MRG\_MyISAM** engine, is a collection of identical **MyISAM** tables that can be used as one. “Identical” means that all tables have identical column data types and index information. You cannot merge **MyISAM** tables in which the columns are listed in a different order, do not have exactly the same data types in corresponding columns, or have the indexes in different order. However, any or all of the **MyISAM** tables can be compressed with [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack). See [Section 4.6.6, “myisampack — Generate Compressed, Read-Only MyISAM Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack). Differences between tables such as these do not matter:

Names of corresponding columns and indexes can differ.

Comments for tables, columns, and indexes can differ.

Table options such as **AVG\_ROW\_LENGTH**, **MAX\_ROWS**, or **PACK\_KEYS** can differ.

An alternative to a **MERGE** table is a partitioned table, which stores partitions of a single table in separate files and enables some operations to be performed more efficiently. For more information, see [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html).

When you create a **MERGE** table, MySQL creates a .MRG file on disk that contains the names of the underlying **MyISAM** tables that should be used as one. The table format of the **MERGE** table is stored in the MySQL data dictionary. The underlying tables do not have to be in the same database as the **MERGE** table.

You can use [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) on **MERGE** tables. You must have [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privileges on the **MyISAM** tables that you map to a **MERGE** table.

**Note**

The use of **MERGE** tables entails the following security issue: If a user has access to **MyISAM** table ***t***, that user can create a **MERGE** table ***m*** that accesses ***t***. However, if the user's privileges on ***t*** are subsequently revoked, the user can continue to access ***t*** by doing so through ***m***.

Use of [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) with a **MERGE** table drops only the **MERGE** specification. The underlying tables are not affected.

To create a **MERGE** table, you must specify a **UNION=(*list-of-tables*)** option that indicates which **MyISAM** tables to use. You can optionally specify an **INSERT\_METHOD** option to control how inserts into the **MERGE** table take place. Use a value of **FIRST** or **LAST** to cause inserts to be made in the first or last underlying table, respectively. If you specify no **INSERT\_METHOD** option or if you specify it with a value of **NO**, inserts into the **MERGE** table are not permitted and attempts to do so result in an error.

The following example shows how to create a **MERGE** table:

mysql> **CREATE TABLE t1 (**

-> **a INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

-> **message CHAR(20)) ENGINE=MyISAM;**

mysql> **CREATE TABLE t2 (**

-> **a INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

-> **message CHAR(20)) ENGINE=MyISAM;**

mysql> **INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');**

mysql> **INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');**

mysql> **CREATE TABLE total (**

-> **a INT NOT NULL AUTO\_INCREMENT,**

-> **message CHAR(20), INDEX(a))**

-> **ENGINE=MERGE UNION=(t1,t2) INSERT\_METHOD=LAST;**

Column **a** is indexed as a **PRIMARY KEY** in the underlying **MyISAM** tables, but not in the **MERGE** table. There it is indexed but not as a **PRIMARY KEY** because a **MERGE** table cannot enforce uniqueness over the set of underlying tables. (Similarly, a column with a **UNIQUE** index in the underlying tables should be indexed in the **MERGE** table but not as a **UNIQUE** index.)

After creating the **MERGE** table, you can use it to issue queries that operate on the group of tables as a whole:

mysql> **SELECT \* FROM total;**

+---+---------+

| a | message |

+---+---------+

| 1 | Testing |

| 2 | table |

| 3 | t1 |

| 1 | Testing |

| 2 | table |

| 3 | t2 |

+---+---------+

To remap a **MERGE** table to a different collection of **MyISAM** tables, you can use one of the following methods:

**DROP** the **MERGE** table and re-create it.

Use **ALTER TABLE *tbl\_name* UNION=(...)** to change the list of underlying tables.

It is also possible to use **ALTER TABLE ... UNION=()** (that is, with an empty [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) clause) to remove all of the underlying tables. However, in this case, the table is effectively empty and inserts fail because there is no underlying table to take new rows. Such a table might be useful as a template for creating new **MERGE** tables with [**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like).

The underlying table definitions and indexes must conform closely to the definition of the **MERGE** table. Conformance is checked when a table that is part of a **MERGE** table is opened, not when the **MERGE** table is created. If any table fails the conformance checks, the operation that triggered the opening of the table fails. This means that changes to the definitions of tables within a **MERGE** may cause a failure when the **MERGE** table is accessed. The conformance checks applied to each table are:

The underlying table and the **MERGE** table must have the same number of columns.

The column order in the underlying table and the **MERGE** table must match.

Additionally, the specification for each corresponding column in the parent **MERGE** table and the underlying tables are compared and must satisfy these checks:

The column type in the underlying table and the **MERGE** table must be equal.

The column length in the underlying table and the **MERGE** table must be equal.

The column of the underlying table and the **MERGE** table can be **NULL**.

The underlying table must have at least as many indexes as the **MERGE** table. The underlying table may have more indexes than the **MERGE** table, but cannot have fewer.

**Note**

A known issue exists where indexes on the same columns must be in identical order, in both the **MERGE** table and the underlying **MyISAM** table. See Bug #33653.

Each index must satisfy these checks:

The index type of the underlying table and the **MERGE** table must be the same.

The number of index parts (that is, multiple columns within a compound index) in the index definition for the underlying table and the **MERGE** table must be the same.

For each index part:

Index part lengths must be equal.

Index part types must be equal.

Index part languages must be equal.

Check whether index parts can be **NULL**.

If a **MERGE** table cannot be opened or used because of a problem with an underlying table, [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) displays information about which table caused the problem.

### Additional Resources

A forum dedicated to the **MERGE** storage engine is available at <https://forums.mysql.com/list.php?93>.

### 16.7.1 MERGE Table Advantages and Disadvantages

**MERGE** tables can help you solve the following problems:

Easily manage a set of log tables. For example, you can put data from different months into separate tables, compress some of them with [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack), and then create a **MERGE** table to use them as one.

Obtain more speed. You can split a large read-only table based on some criteria, and then put individual tables on different disks. A **MERGE** table structured this way could be much faster than using a single large table.

Perform more efficient searches. If you know exactly what you are looking for, you can search in just one of the underlying tables for some queries and use a **MERGE** table for others. You can even have many different **MERGE** tables that use overlapping sets of tables.

Perform more efficient repairs. It is easier to repair individual smaller tables that are mapped to a **MERGE** table than to repair a single large table.

Instantly map many tables as one. A **MERGE** table need not maintain an index of its own because it uses the indexes of the individual tables. As a result, **MERGE** table collections are very fast to create or remap. (You must still specify the index definitions when you create a **MERGE** table, even though no indexes are created.)

If you have a set of tables from which you create a large table on demand, you can instead create a **MERGE** table from them on demand. This is much faster and saves a lot of disk space.

Exceed the file size limit for the operating system. Each **MyISAM** table is bound by this limit, but a collection of **MyISAM** tables is not.

You can create an alias or synonym for a **MyISAM** table by defining a **MERGE** table that maps to that single table. There should be no really notable performance impact from doing this (only a couple of indirect calls and **memcpy()** calls for each read).

The disadvantages of **MERGE** tables are:

You can use only identical **MyISAM** tables for a **MERGE** table.

Some **MyISAM** features are unavailable in **MERGE** tables. For example, you cannot create **FULLTEXT** indexes on **MERGE** tables. (You can create **FULLTEXT** indexes on the underlying **MyISAM** tables, but you cannot search the **MERGE** table with a full-text search.)

If the **MERGE** table is nontemporary, all underlying **MyISAM** tables must be nontemporary. If the **MERGE** table is temporary, the **MyISAM** tables can be any mix of temporary and nontemporary.

**MERGE** tables use more file descriptors than **MyISAM** tables. If 10 clients are using a **MERGE** table that maps to 10 tables, the server uses (10 × 10) + 10 file descriptors. (10 data file descriptors for each of the 10 clients, and 10 index file descriptors shared among the clients.)

Index reads are slower. When you read an index, the **MERGE** storage engine needs to issue a read on all underlying tables to check which one most closely matches a given index value. To read the next index value, the **MERGE** storage engine needs to search the read buffers to find the next value. Only when one index buffer is used up does the storage engine need to read the next index block. This makes **MERGE** indexes much slower on [**eq\_ref**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_eq_ref) searches, but not much slower on [**ref**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_ref) searches. For more information about [**eq\_ref**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_eq_ref) and [**ref**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_ref), see [Section 13.8.2, “EXPLAIN Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain).

### 16.7.2 MERGE Table Problems

The following are known problems with **MERGE** tables:

In versions of MySQL Server prior to 5.1.23, it was possible to create temporary merge tables with nontemporary child MyISAM tables.

From versions 5.1.23, MERGE children were locked through the parent table. If the parent was temporary, it was not locked and so the children were not locked either. Parallel use of the MyISAM tables corrupted them.

If you use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to change a **MERGE** table to another storage engine, the mapping to the underlying tables is lost. Instead, the rows from the underlying **MyISAM** tables are copied into the altered table, which then uses the specified storage engine.

The **INSERT\_METHOD** table option for a **MERGE** table indicates which underlying **MyISAM** table to use for inserts into the **MERGE** table. However, use of the **AUTO\_INCREMENT** table option for that **MyISAM** table has no effect for inserts into the **MERGE** table until at least one row has been inserted directly into the **MyISAM** table.

A **MERGE** table cannot maintain uniqueness constraints over the entire table. When you perform an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), the data goes into the first or last **MyISAM** table (as determined by the **INSERT\_METHOD** option). MySQL ensures that unique key values remain unique within that **MyISAM** table, but not over all the underlying tables in the collection.

Because the **MERGE** engine cannot enforce uniqueness over the set of underlying tables, [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) does not work as expected. The two key facts are:

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) can detect unique key violations only in the underlying table to which it is going to write (which is determined by the **INSERT\_METHOD** option). This differs from violations in the **MERGE** table itself.

If [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) detects a unique key violation, it changes only the corresponding row in the underlying table it is writing to; that is, the first or last table, as determined by the **INSERT\_METHOD** option.

Similar considerations apply for [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

**MERGE** tables do not support partitioning. That is, you cannot partition a **MERGE** table, nor can any of a **MERGE** table's underlying **MyISAM** tables be partitioned.

You should not use [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) without a **WHERE** clause, or [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) on any of the tables that are mapped into an open **MERGE** table. If you do so, the **MERGE** table may still refer to the original table and yield unexpected results. To work around this problem, ensure that no **MERGE** tables remain open by issuing a [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) statement prior to performing any of the named operations.

The unexpected results include the possibility that the operation on the **MERGE** table reports table corruption. If this occurs after one of the named operations on the underlying **MyISAM** tables, the corruption message is spurious. To deal with this, issue a [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) statement after modifying the **MyISAM** tables.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) on a table that is in use by a **MERGE** table does not work on Windows because the **MERGE** storage engine's table mapping is hidden from the upper layer of MySQL. Windows does not permit open files to be deleted, so you first must flush all **MERGE** tables (with [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables)) or drop the **MERGE** table before dropping the table.

The definition of the **MyISAM** tables and the **MERGE** table are checked when the tables are accessed (for example, as part of a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement). The checks ensure that the definitions of the tables and the parent **MERGE** table definition match by comparing column order, types, sizes and associated indexes. If there is a difference between the tables, an error is returned and the statement fails. Because these checks take place when the tables are opened, any changes to the definition of a single table, including column changes, column ordering, and engine alterations cause the statement to fail.

The order of indexes in the **MERGE** table and its underlying tables should be the same. If you use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to add a **UNIQUE** index to a table used in a **MERGE** table, and then use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to add a nonunique index on the **MERGE** table, the index ordering is different for the tables if there was already a nonunique index in the underlying table. (This happens because [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) puts **UNIQUE** indexes before nonunique indexes to facilitate rapid detection of duplicate keys.) Consequently, queries on tables with such indexes may return unexpected results.

If you encounter an error message similar to ERROR 1017 (HY000): Can't find file: '***tbl\_name***.MRG' (errno: 2), it generally indicates that some of the underlying tables do not use the **MyISAM** storage engine. Confirm that all of these tables are **MyISAM**.

The maximum number of rows in a **MERGE** table is 264 (~1.844E+19; the same as for a **MyISAM** table). It is not possible to merge multiple **MyISAM** tables into a single **MERGE** table that would have more than this number of rows.

Use of underlying **MyISAM** tables of differing row formats with a parent **MERGE** table is currently known to fail. See Bug #32364.

You cannot change the union list of a nontemporary **MERGE** table when [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is in effect. The following does not work:

CREATE TABLE m1 ... ENGINE=MRG\_MYISAM ...;

LOCK TABLES t1 WRITE, t2 WRITE, m1 WRITE;

ALTER TABLE m1 ... UNION=(t1,t2) ...;

However, you can do this with a temporary **MERGE** table.

You cannot create a **MERGE** table with **CREATE ... SELECT**, neither as a temporary **MERGE** table, nor as a nontemporary **MERGE** table. For example:

CREATE TABLE m1 ... ENGINE=MRG\_MYISAM ... SELECT ...;

Attempts to do this result in an error: ***tbl\_name*** is not **BASE TABLE**.

In some cases, differing **PACK\_KEYS** table option values among the **MERGE** and underlying tables cause unexpected results if the underlying tables contain **CHAR** or **BINARY** columns. As a workaround, use **ALTER TABLE** to ensure that all involved tables have the same **PACK\_KEYS** value. (Bug #50646)

## 16.8 The FEDERATED Storage Engine

[16.8.1 FEDERATED Storage Engine Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-description)

[16.8.2 How to Create FEDERATED Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-create)

[16.8.3 FEDERATED Storage Engine Notes and Tips](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-usagenotes)

[16.8.4 FEDERATED Storage Engine Resources](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-storage-engine-resources)

The **FEDERATED** storage engine lets you access data from a remote MySQL database without using replication or cluster technology. Querying a local **FEDERATED** table automatically pulls the data from the remote (federated) tables. No data is stored on the local tables.

To include the **FEDERATED** storage engine if you build MySQL from source, invoke **CMake** with the [-DWITH\_FEDERATED\_STORAGE\_ENGINE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_storage_engine_options) option.

The **FEDERATED** storage engine is not enabled by default in the running server; to enable **FEDERATED**, you must start the MySQL server binary using the --federated option.

To examine the source for the **FEDERATED** engine, look in the storage/federated directory of a MySQL source distribution.

### 16.8.1 FEDERATED Storage Engine Overview

When you create a table using one of the standard storage engines (such as **MyISAM**, **CSV** or **InnoDB**), the table consists of the table definition and the associated data. When you create a **FEDERATED** table, the table definition is the same, but the physical storage of the data is handled on a remote server.

A **FEDERATED** table consists of two elements:

A remote server with a database table, which in turn consists of the table definition (stored in the MySQL data dictionary) and the associated table. The table type of the remote table may be any type supported by the remote **mysqld** server, including **MyISAM** or **InnoDB**.

A local server with a database table, where the table definition matches that of the corresponding table on the remote server. The table definition is stored in the data dictionary. There is no data file on the local server. Instead, the table definition includes a connection string that points to the remote table.

When executing queries and statements on a **FEDERATED** table on the local server, the operations that would normally insert, update or delete information from a local data file are instead sent to the remote server for execution, where they update the data file on the remote server or return matching rows from the remote server.

The basic structure of a **FEDERATED** table setup is shown in [Figure 16.2, “FEDERATED Table Structure”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#figure-se-federated-structure).

**Figure 16.2 FEDERATED Table Structure**

When a client issues an SQL statement that refers to a **FEDERATED** table, the flow of information between the local server (where the SQL statement is executed) and the remote server (where the data is physically stored) is as follows:

The storage engine looks through each column that the **FEDERATED** table has and constructs an appropriate SQL statement that refers to the remote table.

The statement is sent to the remote server using the MySQL client API.

The remote server processes the statement and the local server retrieves any result that the statement produces (an affected-rows count or a result set).

If the statement produces a result set, each column is converted to internal storage engine format that the **FEDERATED** engine expects and can use to display the result to the client that issued the original statement.

The local server communicates with the remote server using MySQL client C API functions. It invokes [**mysql\_real\_query()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-query.html) to send the statement. To read a result set, it uses [**mysql\_store\_result()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-store-result.html) and fetches rows one at a time using [**mysql\_fetch\_row()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-fetch-row.html).

### 16.8.2 How to Create FEDERATED Tables

[16.8.2.1 Creating a FEDERATED Table Using CONNECTION](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-create-connection)

[16.8.2.2 Creating a FEDERATED Table Using CREATE SERVER](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-create-server)

To create a **FEDERATED** table you should follow these steps:

Create the table on the remote server. Alternatively, make a note of the table definition of an existing table, perhaps using the [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) statement.

Create the table on the local server with an identical table definition, but adding the connection information that links the local table to the remote table.

For example, you could create the following table on the remote server:

CREATE TABLE test\_table (

id INT(20) NOT NULL AUTO\_INCREMENT,

name VARCHAR(32) NOT NULL DEFAULT '',

other INT(20) NOT NULL DEFAULT '0',

PRIMARY KEY (id),

INDEX name (name),

INDEX other\_key (other)

)

ENGINE=MyISAM

DEFAULT CHARSET=utf8mb4;

To create the local table that is federated to the remote table, there are two options available. You can either create the local table and specify the connection string (containing the server name, login, password) to be used to connect to the remote table using the **CONNECTION**, or you can use an existing connection that you have previously created using the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement.

**Important**

When you create the local table it must have an identical field definition to the remote table.

**Note**

You can improve the performance of a **FEDERATED** table by adding indexes to the table on the host. The optimization occurs because the query sent to the remote server includes the contents of the **WHERE** clause and is sent to the remote server and subsequently executed locally. This reduces the network traffic that would otherwise request the entire table from the server for local processing.

#### 16.8.2.1 Creating a FEDERATED Table Using CONNECTION

To use the first method, you must specify the **CONNECTION** string after the engine type in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. For example:

CREATE TABLE federated\_table (

id INT(20) NOT NULL AUTO\_INCREMENT,

name VARCHAR(32) NOT NULL DEFAULT '',

other INT(20) NOT NULL DEFAULT '0',

PRIMARY KEY (id),

INDEX name (name),

INDEX other\_key (other)

)

ENGINE=FEDERATED

DEFAULT CHARSET=utf8mb4

CONNECTION='mysql://fed\_user@remote\_host:9306/federated/test\_table';

**Note**

**CONNECTION** replaces the **COMMENT** used in some previous versions of MySQL.

The **CONNECTION** string contains the information required to connect to the remote server containing the table in which the data physically resides. The connection string specifies the server name, login credentials, port number and database/table information. In the example, the remote table is on the server **remote\_host**, using port 9306. The name and port number should match the host name (or IP address) and port number of the remote MySQL server instance you want to use as your remote table.

The format of the connection string is as follows:

***scheme***://***user\_name***[:***password***]@***host\_name***[:***port\_num***]/***db\_name***/***tbl\_name***

Where:

***scheme***: A recognized connection protocol. Only **mysql** is supported as the ***scheme*** value at this point.

***user\_name***: The user name for the connection. This user must have been created on the remote server, and must have suitable privileges to perform the required actions ([**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and so forth) on the remote table.

***password***: (Optional) The corresponding password for ***user\_name***.

***host\_name***: The host name or IP address of the remote server.

***port\_num***: (Optional) The port number for the remote server. The default is 3306.

***db\_name***: The name of the database holding the remote table.

***tbl\_name***: The name of the remote table. The name of the local and the remote table do not have to match.

Sample connection strings:

CONNECTION='mysql://username:password@hostname:port/database/tablename'

CONNECTION='mysql://username@hostname/database/tablename'

CONNECTION='mysql://username:password@hostname/database/tablename'

#### 16.8.2.2 Creating a FEDERATED Table Using CREATE SERVER

If you are creating a number of **FEDERATED** tables on the same server, or if you want to simplify the process of creating **FEDERATED** tables, you can use the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement to define the server connection parameters, just as you would with the **CONNECTION** string.

The format of the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement is:

CREATE SERVER

***server\_name***

FOREIGN DATA WRAPPER ***wrapper\_name***

OPTIONS (***option*** [, ***option***] ...)

The ***server\_name*** is used in the connection string when creating a new **FEDERATED** table.

For example, to create a server connection identical to the **CONNECTION** string:

CONNECTION='mysql://fed\_user@remote\_host:9306/federated/test\_table';

You would use the following statement:

CREATE SERVER fedlink

FOREIGN DATA WRAPPER mysql

OPTIONS (USER 'fed\_user', HOST 'remote\_host', PORT 9306, DATABASE 'federated');

To create a **FEDERATED** table that uses this connection, you still use the **CONNECTION** keyword, but specify the name you used in the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement.

CREATE TABLE test\_table (

id INT(20) NOT NULL AUTO\_INCREMENT,

name VARCHAR(32) NOT NULL DEFAULT '',

other INT(20) NOT NULL DEFAULT '0',

PRIMARY KEY (id),

INDEX name (name),

INDEX other\_key (other)

)

ENGINE=FEDERATED

DEFAULT CHARSET=utf8mb4

CONNECTION='fedlink/test\_table';

The connection name in this example contains the name of the connection (**fedlink**) and the name of the table (**test\_table**) to link to, separated by a slash. If you specify only the connection name without a table name, the table name of the local table is used instead.

For more information on [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server), see [Section 13.1.18, “CREATE SERVER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server).

The [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement accepts the same arguments as the **CONNECTION** string. The [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement updates the rows in the **mysql.servers** table. See the following table for information on the correspondence between parameters in a connection string, options in the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement, and the columns in the **mysql.servers** table. For reference, the format of the **CONNECTION** string is as follows:

***scheme***://***user\_name***[:***password***]@***host\_name***[:***port\_num***]/***db\_name***/***tbl\_name***

|  |  |  |  |
| --- | --- | --- | --- |
| Description | CONNECTION string | [CREATE SERVER](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) option | mysql.servers column |
| Connection scheme | ***scheme*** | **wrapper\_name** | **Wrapper** |
| Remote user | ***user\_name*** | **USER** | **Username** |
| Remote password | ***password*** | **PASSWORD** | **Password** |
| Remote host | ***host\_name*** | **HOST** | **Host** |
| Remote port | ***port\_num*** | **PORT** | **Port** |
| Remote database | ***db\_name*** | **DATABASE** | **Db** |

### 16.8.3 FEDERATED Storage Engine Notes and Tips

You should be aware of the following points when using the **FEDERATED** storage engine:

**FEDERATED** tables may be replicated to other replicas, but you must ensure that the replica servers are able to use the user/password combination that is defined in the **CONNECTION** string (or the row in the **mysql.servers** table) to connect to the remote server.

The following items indicate features that the **FEDERATED** storage engine does and does not support:

The remote server must be a MySQL server.

The remote table that a **FEDERATED** table points to must exist before you try to access the table through the **FEDERATED** table.

It is possible for one **FEDERATED** table to point to another, but you must be careful not to create a loop.

A **FEDERATED** table does not support indexes in the usual sense; because access to the table data is handled remotely, it is actually the remote table that makes use of indexes. This means that, for a query that cannot use any indexes and so requires a full table scan, the server fetches all rows from the remote table and filters them locally. This occurs regardless of any **WHERE** or **LIMIT** used with this [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement; these clauses are applied locally to the returned rows.

Queries that fail to use indexes can thus cause poor performance and network overload. In addition, since returned rows must be stored in memory, such a query can also lead to the local server swapping, or even hanging.

Care should be taken when creating a **FEDERATED** table since the index definition from an equivalent **MyISAM** or other table may not be supported. For example, creating a **FEDERATED** table fails if the table uses an index prefix on any [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. The following definition using **MyISAM** is valid:

CREATE TABLE `T1`(`A` VARCHAR(100),UNIQUE KEY(`A`(30))) ENGINE=MYISAM;

The key prefix in this example is incompatible with the **FEDERATED** engine, and the equivalent statement fails:

CREATE TABLE `T1`(`A` VARCHAR(100),UNIQUE KEY(`A`(30))) ENGINE=FEDERATED

CONNECTION='MYSQL://127.0.0.1:3306/TEST/T1';

If possible, you should try to separate the column and index definition when creating tables on both the remote server and the local server to avoid these index issues.

Internally, the implementation uses [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), but not [**HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler).

The **FEDERATED** storage engine supports [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), and indexes. It does not support [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or any Data Definition Language statements that directly affect the structure of the table, other than [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table). The current implementation does not use prepared statements.

**FEDERATED** accepts [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements, but if a duplicate-key violation occurs, the statement fails with an error.

Transactions are not supported.

**FEDERATED** performs bulk-insert handling such that multiple rows are sent to the remote table in a batch, which improves performance. Also, if the remote table is transactional, it enables the remote storage engine to perform statement rollback properly should an error occur. This capability has the following limitations:

The size of the insert cannot exceed the maximum packet size between servers. If the insert exceeds this size, it is broken into multiple packets and the rollback problem can occur.

Bulk-insert handling does not occur for [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

There is no way for the **FEDERATED** engine to know if the remote table has changed. The reason for this is that this table must work like a data file that would never be written to by anything other than the database system. The integrity of the data in the local table could be breached if there was any change to the remote database.

When using a **CONNECTION** string, you cannot use an '@' character in the password. You can get round this limitation by using the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement to create a server connection.

The [**insert\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_insert_id) and [**timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_timestamp) options are not propagated to the data provider.

Any [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement issued against a **FEDERATED** table drops only the local table, not the remote table.

User-defined partitioning is not supported for **FEDERATED** tables.

### 16.8.4 FEDERATED Storage Engine Resources

The following additional resources are available for the **FEDERATED** storage engine:

A forum dedicated to the **FEDERATED** storage engine is available at <https://forums.mysql.com/list.php?105>.

## 16.9 The EXAMPLE Storage Engine

The **EXAMPLE** storage engine is a stub engine that does nothing. Its purpose is to serve as an example in the MySQL source code that illustrates how to begin writing new storage engines. As such, it is primarily of interest to developers.

To enable the **EXAMPLE** storage engine if you build MySQL from source, invoke **CMake** with the [-DWITH\_EXAMPLE\_STORAGE\_ENGINE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_storage_engine_options) option.

To examine the source for the **EXAMPLE** engine, look in the storage/example directory of a MySQL source distribution.

When you create an **EXAMPLE** table, no files are created. No data can be stored into the table. Retrievals return an empty result.

mysql> **CREATE TABLE test (i INT) ENGINE = EXAMPLE;**

Query OK, 0 rows affected (0.78 sec)

mysql> **INSERT INTO test VALUES(1),(2),(3);**

ERROR 1031 (HY000): Table storage engine for 'test' doesn't »

have this option

mysql> **SELECT \* FROM test;**

Empty set (0.31 sec)

The **EXAMPLE** storage engine does not support indexing.

The **EXAMPLE** storage engine does not support partitioning.

## 16.10 其他存储引擎

其他存储引擎可能来自第三方和使用了自定义存储引擎接口的社区成员。

第三方引擎不受MySQL支持。对于进一步的信息、文档、安装指南、错误报告或对这些引擎的任何帮助或协助，请直接联系引擎的开发者。

关于开发可与可插拔存储引擎架构一起使用的客户存储引擎的更多信息，请参见MySQL内部。编写自定义存储引擎

## 16.11 mysql存储引擎架构概览

[16.11.1 Pluggable Storage Engine Architecture](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#pluggable-storage)

[16.11.2 The Common Database Server Layer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#pluggable-storage-common-layer)

MySQL的可插拔存储引擎架构使数据库专业人员能够为特定的应用需求选择一个专门的存储引擎，同时完全不需要管理任何特定的应用编码要求。MySQL服务器架构将应用程序员和DBA与存储层面的所有低级实施细节隔离开来，提供了一个一致且简单的应用模型和API。因此，尽管不同的存储引擎有不同的能力，但应用程序被屏蔽在这些差异之外。

MySQL的可插拔存储引擎架构如图16.3所示，"具有可插拔存储引擎的MySQL架构"。**Figure 16.3 MySQL Architecture with Pluggable Storage Engines**

可插拔的存储引擎架构提供了一套标准的管理和支持服务，这些服务在所有底层存储引擎中是通用的。存储引擎本身是数据库服务器的组件，实际上是对在物理服务器层面上维护的底层数据进行操作。

这种高效和模块化的架构为那些希望专门针对特定应用需求的人提供了巨大的好处，例如数据仓库、交易处理或高可用性情况，同时享有利用一套独立于任何一个存储引擎的接口和服务的优势。

应用程序员和DBA通过高于存储引擎的连接器API和服务层与MySQL数据库互动。如果应用程序的变化带来了要求改变底层存储引擎的需求，或者需要增加一个或多个存储引擎来支持新的需求，则不需要进行重大的编码或流程改变来使事情顺利进行。MySQL服务器架构通过提供适用于各种存储引擎的一致且易于使用的API，使应用程序免受存储引擎的底层复杂性的影响。

### 16.11.1 Pluggable Storage Engine Architecture

MySQL Server uses a pluggable storage engine architecture that enables storage engines to be loaded into and unloaded from a running MySQL server.

***Plugging in a Storage Engine***

Before a storage engine can be used, the storage engine plugin shared library must be loaded into MySQL using the [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement. For example, if the **EXAMPLE** engine plugin is named **example** and the shared library is named ha\_example.so, you load it with the following statement:

INSTALL PLUGIN example SONAME 'ha\_example.so';

To install a pluggable storage engine, the plugin file must be located in the MySQL plugin directory, and the user issuing the [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement must have [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **mysql.plugin** table.

The shared library must be located in the MySQL server plugin directory, the location of which is given by the [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable.

***Unplugging a Storage Engine***

To unplug a storage engine, use the [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) statement:

UNINSTALL PLUGIN example;

If you unplug a storage engine that is needed by existing tables, those tables become inaccessible, but are still present on disk (where applicable). Ensure that there are no tables using a storage engine before you unplug the storage engine.

### 16.11.2 The Common Database Server Layer

MySQL可插拔存储引擎是MySQL数据库服务器中的一个组件，负责执行数据库的实际数据I/O操作，以及启用和执行针对特定应用需求的某些功能集。使用特定存储引擎的一个主要好处是，你只被交付给特定应用所需的功能，因此你在数据库中的系统开销较少，最终结果是更有效和更高的数据库性能。这也是MySQL一直以来被称为具有如此高的性能的原因之一，在行业标准的基准中与专有的单片机数据库相匹配或击败。

从技术角度来看，存储引擎中有哪些独特的支持性基础设施组件？一些关键的功能差异包括。Concurrency: Some applications have more granular lock requirements (such as row-level locks) than others. Choosing the right locking strategy can reduce overhead and therefore improve overall performance. This area also includes support for capabilities such as multi-version concurrency control or “snapshot” read.

Transaction Support: Not every application needs transactions, but for those that do, there are very well defined requirements such as ACID compliance and more.

Referential Integrity: The need to have the server enforce relational database referential integrity through DDL defined foreign keys.

Physical Storage: This involves everything from the overall page size for tables and indexes as well as the format used for storing data to physical disk.

Index Support: Different application scenarios tend to benefit from different index strategies. Each storage engine generally has its own indexing methods, although some (such as B-tree indexes) are common to nearly all engines.

Memory Caches: Different applications respond better to some memory caching strategies than others, so although some memory caches are common to all storage engines (such as those used for user connections), others are uniquely defined only when a particular storage engine is put in play.

Performance Aids: This includes multiple I/O threads for parallel operations, thread concurrency, database checkpointing, bulk insert handling, and more.

Miscellaneous Target Features: This may include support for geospatial operations, security restrictions for certain data manipulation operations, and other similar features.

Each set of the pluggable storage engine infrastructure components are designed to offer a selective set of benefits for a particular application. Conversely, avoiding a set of component features helps reduce unnecessary overhead. It stands to reason that understanding a particular application's set of requirements and selecting the proper MySQL storage engine can have a dramatic impact on overall system efficiency and performance.

# 复制

复制使来自一个MySQL数据库服务器（称为源）的数据被复制到一个或多个MySQL数据库服务器（称为副本）。复制在默认情况下是异步的；复制体不需要永久连接以接收来自源的更新。根据配置，你可以复制所有数据库、选定的数据库，甚至是数据库中选定的表。

在MySQL中复制的优点包括。

扩展解决方案--在多个副本中分散负载以提高性能。在这种环境中，所有的写入和更新必须在源服务器上进行。然而，读取可能发生在一个或多个副本上。这种模式可以提高写的性能（因为源是专门用于更新的），同时在越来越多的复制中大幅提高读的速度。

数据安全--因为副本可以暂停复制过程，所以有可能在副本上运行备份服务而不破坏相应的源数据。

分析--可以在源数据上创建实时数据，而信息的分析可以在副本上进行，而不影响源数据的性能。

长距离数据分发--你可以使用复制来创建一个本地数据副本，供远程站点使用，而不需要永久访问源。

关于如何在这种情况下使用复制的信息，见第17.4节 "复制解决方案"。

MySQL 8.0支持不同的复制方法。传统的方法是基于复制源的二进制日志的事件，并要求日志文件和其中的位置在源和副本之间同步。较新的基于全局事务标识符（GTIDs）的方法是事务性的，因此不需要处理日志文件或这些文件中的位置，这大大简化了许多常见的复制任务。使用GTIDs的复制保证了源和副本之间的一致性，只要在源上提交的所有事务也被应用到副本上。关于MySQL中GTID和基于GTID的复制的更多信息，见第17.1.3节，"使用全局事务标识符的复制"。关于使用基于二进制日志文件位置的复制的信息，见第17.1节，"配置复制"。

MySQL中的复制支持不同类型的同步。原始的同步类型是单向的、异步的复制，其中一个服务器作为源，而一个或多个其他服务器作为副本。这与NDB Cluster的特点--同步复制形成对比（见第23章，MySQL NDB Cluster 8.0）。在MySQL 8.0中，除了内置的异步复制外，还支持半同步复制。通过半同步复制，在返回到执行事务的会话之前，在源块上执行的提交，直到至少一个副本确认它已经收到并记录了该事务的事件；参见第17.4.10节，"半同步复制"。MySQL 8.0还支持延迟复制，即一个副本故意落后于源至少一个指定的时间量；见第17.4.11节 "延迟复制"。对于需要同步复制的情况，请使用NDB Cluster（见第23章，MySQL NDB Cluster 8.0）。

有许多解决方案可用于设置服务器之间的复制，而使用的最佳方法取决于数据的存在和你使用的引擎类型。有关可用选项的更多信息，请参见第17.1.2节，"设置基于二进制日志文件位置的复制"。

有两种核心类型的复制格式，基于语句的复制（SBR）和基于行的复制（RBR），前者复制整个SQL语句，后者只复制改变的行。你也可以使用第三个品种，即混合型复制（MBR）。关于不同复制格式的更多信息，请参见第17.2.1节，"复制格式"。

复制是通过一些不同的选项和变量来控制的。更多信息，请参见第17.1.6节，"复制和二进制日志选项和变量"。如第17.3节 "复制安全性 "所述，额外的安全措施可以应用于复制拓扑结构。

你可以使用复制来解决一些不同的问题，包括性能、支持不同数据库的备份，以及作为一个更大的解决方案的一部分来缓解系统故障。关于如何解决这些问题的信息，请参阅第17.4节 "复制解决方案"

关于复制过程中如何处理不同数据类型和语句的说明和提示，包括复制特性、版本兼容性、升级和潜在问题及其解决的细节，见第17.5节 "复制说明和提示"。关于那些刚接触MySQL复制的人经常问到的一些问题的答案，见A.14节，"MySQL 8.0常见问题。复制"。

关于复制的实现、复制如何工作、二进制日志的过程和内容、后台线程以及用于决定如何记录和复制语句的规则的详细信息，见第17.2节 "复制实现"。

**17.1 Configuring Replication**

[17.1.1 Binary Log File Position Based Replication Configuration Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#binlog-replication-configuration-overview)

[17.1.2 Setting Up Binary Log File Position Based Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto)

[17.1.3 Replication with Global Transaction Identifiers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids)

[17.1.4 Changing GTID Mode on Online Servers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online)

[17.1.5 MySQL Multi-Source Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source)

[17.1.6 Replication and Binary Logging Options and Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options)

[17.1.7 Common Replication Administration Tasks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration)

This section describes how to configure the different types of replication available in MySQL and includes the setup and configuration required for a replication environment, including step-by-step instructions for creating a new replication environment. The major components of this section are:

For a guide to setting up two or more servers for replication using binary log file positions, [Section 17.1.2, “Setting Up Binary Log File Position Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto), deals with the configuration of the servers and provides methods for copying data between the source and replicas.

For a guide to setting up two or more servers for replication using GTID transactions, [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids), deals with the configuration of the servers.

Events in the binary log are recorded using a number of formats. These are referred to as statement-based replication (SBR) or row-based replication (RBR). A third type, mixed-format replication (MIXED), uses SBR or RBR replication automatically to take advantage of the benefits of both SBR and RBR formats when appropriate. The different formats are discussed in [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).

Detailed information on the different configuration options and variables that apply to replication is provided in [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options).

Once started, the replication process should require little administration or monitoring. However, for advice on common tasks that you may want to execute, see [Section 17.1.7, “Common Replication Administration Tasks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration).

**17.1.1 Binary Log File Position Based Replication Configuration Overview**

本节描述了基于二进制日志文件位置方法的MySQL服务器之间的复制，其中作为源（数据库变化发生的地方）运行的MySQL实例将更新和变化作为 "事件 "写入二进制日志。二进制日志中的信息根据被记录的数据库变化，以不同的日志格式存储。复制体被配置为从源头读取二进制日志，并在复制体的本地数据库上执行二进制日志中的事件。

每个副本都收到一份二进制日志全部内容的副本。决定二进制日志中哪些语句应该被执行是副本的责任。除非你另外指定，源的二进制日志中的所有事件都在副本上执行。如果需要，你可以配置副本，使其只处理适用于特定数据库或表的事件。**Important**

You cannot configure the source to log only certain events.

Each replica keeps a record of the binary log coordinates: the file name and position within the file that it has read and processed from the source. This means that multiple replicas can be connected to the source and executing different parts of the same binary log. Because the replicas control this process, individual replicas can be connected and disconnected from the server without affecting the source's operation. Also, because each replica records the current position within the binary log, it is possible for replicas to be disconnected, reconnect and then resume processing.

The source and each replica must be configured with a unique ID (using the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable). In addition, each replica must be configured with information about the source's host name, log file name, and position within that file. These details can be controlled from within a MySQL session using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) on the replica. The details are stored within the replica's connection metadata repository (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)).

**17.1.2 Setting Up Binary Log File Position Based Replication**

[17.1.2.1 Setting the Replication Source Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterbaseconfig)

[17.1.2.2 Setting the Replica Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slavebaseconfig)

[17.1.2.3 Creating a User for Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-repuser)

[17.1.2.4 Obtaining the Replication Source Binary Log Coordinates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus)

[17.1.2.5 Choosing a Method for Data Snapshots](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-snapshot-method)

[17.1.2.6 Setting Up Replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-setup-replicas)

[17.1.2.7 Setting the Source Configuration on the Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slaveinit)

[17.1.2.8 Adding Replicas to a Replication Environment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-additionalslaves)

This section describes how to set up a MySQL server to use binary log file position based replication. There are a number of different methods for setting up replication, and the exact method to use depends on how you are setting up replication, and whether you already have data in the database on the source that you want to replicate.

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

There are some generic tasks that are common to all setups:

On the source, you must ensure that binary logging is enabled, and configure a unique server ID. This might require a server restart. See [Section 17.1.2.1, “Setting the Replication Source Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterbaseconfig).

On each replica that you want to connect to the source, you must configure a unique server ID. This might require a server restart. See [Section 17.1.2.2, “Setting the Replica Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slavebaseconfig).

Optionally, create a separate user for your replicas to use during authentication with the source when reading the binary log for replication. See [Section 17.1.2.3, “Creating a User for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-repuser).

Before creating a data snapshot or starting the replication process, on the source you should record the current position in the binary log. You need this information when configuring the replica so that the replica knows where within the binary log to start executing events. See [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus).

If you already have data on the source and want to use it to synchronize the replica, you need to create a data snapshot to copy the data to the replica. The storage engine you are using has an impact on how you create the snapshot. When you are using [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), you must stop processing statements on the source to obtain a read-lock, then obtain its current binary log coordinates and dump its data, before permitting the source to continue executing statements. If you do not stop the execution of statements, the data dump and the source status information become mismatched, resulting in inconsistent or corrupted databases on the replicas. For more information on replicating a [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) source, see [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus). If you are using [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), you do not need a read-lock and a transaction that is long enough to transfer the data snapshot is sufficient. For more information, see [Section 15.19, “InnoDB and MySQL Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-and-mysql-replication).

Configure the replica with settings for connecting to the source, such as the host name, login credentials, and binary log file name and position. See [Section 17.1.2.7, “Setting the Source Configuration on the Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slaveinit).

Implement replication-specific security measures on the sources and replicas as appropriate for your system. See [Section 17.3, “Replication Security”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-security).

**Note**

Certain steps within the setup process require the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. If you do not have this privilege, it might not be possible to enable replication.

After configuring the basic options, select your scenario:

To set up replication for a fresh installation of a source and replicas that contain no data, see [Section 17.1.2.6.1, “Setting Up Replication with New Source and Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-newservers).

To set up replication of a new source using the data from an existing MySQL server, see [Section 17.1.2.6.2, “Setting Up Replication with Existing Data”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-existingdata).

To add replicas to an existing replication environment, see [Section 17.1.2.8, “Adding Replicas to a Replication Environment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-additionalslaves).

Before administering MySQL replication servers, read this entire chapter and try all statements mentioned in [Section 13.4.1, “SQL Statements for Controlling Source Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-master), and [Section 13.4.2, “SQL Statements for Controlling Replica Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-replica). Also familiarize yourself with the replication startup options described in [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options).

**17.1.2.1 Setting the Replication Source Configuration**

To configure a source to use binary log file position based replication, you must ensure that binary logging is enabled, and establish a unique server ID.

Each server within a replication topology must be configured with a unique server ID, which you can specify using the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable. This server ID is used to identify individual servers within the replication topology, and must be a positive integer between 1 and (232)−1. The default [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value from MySQL 8.0 is 1. You can change the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value dynamically by issuing a statement like this:

SET GLOBAL server\_id = 2;

How you organize and select the server IDs is your choice, so long as each server ID is different from every other server ID in use by any other server in the replication topology. Note that if a value of 0 (which was the default in earlier releases) was set previously for the server ID, you must restart the server to initialize the source with your new nonzero server ID. Otherwise, a server restart is not needed when you change the server ID, unless you make other configuration changes that require it.

Binary logging is required on the source because the binary log is the basis for replicating changes from the source to its replicas. Binary logging is enabled by default (the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable is set to ON). The [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option tells the server what base name to use for binary log files. It is recommended that you specify this option to give the binary log files a non-default base name, so that if the host name changes, you can easily continue to use the same binary log file names (see [Section B.3.7, “Known Issues in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#known-issues)). If binary logging was previously disabled on the source using the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option, you must restart the server without this option to enable it.

**Note**

The following options also have an impact on the source:

For the greatest possible durability and consistency in a replication setup using [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) with transactions, you should use **innodb\_flush\_log\_at\_trx\_commit=1** and **sync\_binlog=1** in the source's my.cnf file.

Ensure that the [**skip\_networking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_skip_networking) system variable is not enabled on the source. If networking has been disabled, the replica cannot communicate with the source and replication fails.

**17.1.2.2 Setting the Replica Configuration**

Each replica must have a unique server ID, as specified by the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable. If you are setting up multiple replicas, each one must have a unique [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value that differs from that of the source and from any of the other replicas. If the replica's server ID is not already set, or the current value conflicts with the value that you have chosen for the source or another replica, you must change it.

The default [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value is 1. You can change the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value dynamically by issuing a statement like this:

SET GLOBAL server\_id = 21;

Note that a value of 0 for the server ID prevents a replica from connecting to a source. If that server ID value (which was the default in earlier releases) was set previously, you must restart the server to initialize the replica with your new nonzero server ID. Otherwise, a server restart is not needed when you change the server ID, unless you make other configuration changes that require it. For example, if binary logging was disabled on the server and you want it enabled for your replica, a server restart is required to enable this.

If you are shutting down the replica server, you can edit the **[mysqld]** section of the configuration file to specify a unique server ID. For example:

[mysqld]

server-id=21

Binary logging is enabled by default on all servers. A replica is not required to have binary logging enabled for replication to take place. However, binary logging on a replica means that the replica's binary log can be used for data backups and crash recovery. Replicas that have binary logging enabled can also be used as part of a more complex replication topology. For example, you might want to set up replication servers using this chained arrangement:

A -> B -> C

Here, **A** serves as the source for the replica **B**, and **B** serves as the source for the replica **C**. For this to work, **B** must be both a source *and* a replica. Updates received from **A** must be logged by **B** to its binary log, in order to be passed on to **C**. In addition to binary logging, this replication topology requires the [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) system variable to be enabled. With replica updates enabled, the replica writes updates that are received from a source and performed by the replica's SQL thread to the replica's own binary log. The [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) system variable is enabled by default.

If you need to disable binary logging or replica update logging on a replica, you can do this by specifying the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) and [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) options for the replica. If you decide to re-enable these features on the replica, remove the relevant options and restart the server.

**17.1.2.3 Creating a User for Replication**

Each replica connects to the source using a MySQL user name and password, so there must be a user account on the source that the replica can use to connect. The user name is specified by the **SOURCE\_USER** | **MASTER\_USER** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) when you set up a replica. Any account can be used for this operation, providing it has been granted the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege. You can choose to create a different account for each replica, or connect to the source using the same account for each replica.

Although you do not have to create an account specifically for replication, you should be aware that the replication user name and password are stored in plain text in the replica's connection metadata repository **mysql.slave\_master\_info** (see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status)). Therefore, you may want to create a separate account that has privileges only for the replication process, to minimize the possibility of compromise to other accounts.

To create a new account, use [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user). To grant this account the privileges required for replication, use the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement. If you create an account solely for the purposes of replication, that account needs only the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege. For example, to set up a new user, **repl**, that can connect for replication from any host within the **example.com** domain, issue these statements on the source:

mysql> **CREATE USER 'repl'@'%.example.com' IDENTIFIED BY '*password*';**

mysql> **GRANT REPLICATION SLAVE ON \*.\* TO 'repl'@'%.example.com';**

See [Section 13.7.1, “Account Management Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#account-management-statements), for more information on statements for manipulation of user accounts.

**Important**

To connect to the source using a user account that authenticates with the **caching\_sha2\_password** plugin, you must either set up a secure connection as described in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections), or enable the unencrypted connection to support password exchange using an RSA key pair. The **caching\_sha2\_password** authentication plugin is the default for new users created from MySQL 8.0 (for details, see [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication)). If the user account that you create or use for replication (as specified by the **MASTER\_USER** option) uses this authentication plugin, and you are not using a secure connection, you must enable RSA key pair-based password exchange for a successful connection.

**17.1.2.4 Obtaining the Replication Source Binary Log Coordinates**

To configure the replica to start the replication process at the correct point, you need to note the source's current coordinates within its binary log.

**Warning**

This procedure uses [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock), which blocks [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) operations for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables.

If you are planning to shut down the source to create a data snapshot, you can optionally skip this procedure and instead store a copy of the binary log index file along with the data snapshot. In that situation, the source creates a new binary log file on restart. The source binary log coordinates where the replica must start the replication process are therefore the start of that new file, which is the next binary log file on the source following after the files that are listed in the copied binary log index file.

To obtain the source binary log coordinates, follow these steps:

Start a session on the source by connecting to it with the command-line client, and flush all tables and block write statements by executing the [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) statement:

mysql> **FLUSH TABLES WITH READ LOCK;**

**Warning**

Leave the client from which you issued the [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) statement running so that the read lock remains in effect. If you exit the client, the lock is released.

In a different session on the source, use the [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) statement to determine the current binary log file name and position:

mysql > **SHOW MASTER STATUS;**

+------------------+----------+--------------+------------------+

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |

+------------------+----------+--------------+------------------+

| mysql-bin.000003 | 73 | test | manual,mysql |

+------------------+----------+--------------+------------------+

The **File** column shows the name of the log file and the **Position** column shows the position within the file. In this example, the binary log file is **mysql-bin.000003** and the position is 73. Record these values. You need them later when you are setting up the replica. They represent the replication coordinates at which the replica should begin processing new updates from the source.

If the source has been running previously with binary logging disabled, the log file name and position values displayed by [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) or [**mysqldump --master-data**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) are empty. In that case, the values that you need to use later when specifying the source's binary log file and position are the empty string (**''**) and **4**.

You now have the information you need to enable the replica to start reading from the source's binary log in the correct place to start replication.

The next step depends on whether you have existing data on the source. Choose one of the following options:

If you have existing data that needs be to synchronized with the replica before you start replication, leave the client running so that the lock remains in place. This prevents any further changes being made, so that the data copied to the replica is in synchrony with the source. Proceed to [Section 17.1.2.5, “Choosing a Method for Data Snapshots”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-snapshot-method).

If you are setting up a new source and replica combination, you can exit the first session to release the read lock. See [Section 17.1.2.6.1, “Setting Up Replication with New Source and Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-newservers) for how to proceed.

**17.1.2.5 Choosing a Method for Data Snapshots**

If the source database contains existing data it is necessary to copy this data to each replica. There are different ways to dump the data from the source database. The following sections describe possible options.

To select the appropriate method of dumping the database, choose between these options:

Use the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) tool to create a dump of all the databases you want to replicate. This is the recommended method, especially when using [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html).

If your database is stored in binary portable files, you can copy the raw data files to a replica. This can be more efficient than using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) and importing the file on each replica, because it skips the overhead of updating indexes as the **INSERT** statements are replayed. With storage engines such as [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) this is not recommended.

Use MySQL Server's clone plugin to transfer all the data from an existing replica to a clone. For instructions to use this method, see [Section 5.6.7.6, “Cloning for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-replication).

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

**17.1.2.5.1 Creating a Data Snapshot Using mysqldump**

To create a snapshot of the data in an existing source database, use the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) tool. Once the data dump has been completed, import this data into the replica before starting the replication process.

The following example dumps all databases to a file named dbdump.db, and includes the [--master-data](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_master-data) option which automatically appends the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement required on the replica to start the replication process:

shell> **mysqldump --all-databases --master-data > dbdump.db**

**Note**

If you do not use [--master-data](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_master-data), then it is necessary to lock all tables in a separate session manually. See [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus).

It is possible to exclude certain databases from the dump using the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) tool. If you want to choose which databases to include in the dump, do not use [--all-databases](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_all-databases). Choose one of these options:

Exclude all the tables in the database using [--ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_ignore-table) option.

Name only those databases which you want dumped using the [--databases](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_databases) option.

**Note**

By default, if GTIDs are in use on the source ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) includes the GTIDs from the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on the source in the dump output to add them to the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set on the replica. If you are dumping only specific databases or tables, it is important to note that the value that is included by [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) includes the GTIDs of all transactions in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on the source, even those that changed suppressed parts of the database, or other databases on the server that were not included in the partial dump. Check the description for mysqldump's --set-gtid-purged option to find the outcome of the default behavior for the MySQL Server versions you are using, and how to change the behavior if this outcome is not suitable for your situation.

For more information, see [Section 4.5.4, “mysqldump — A Database Backup Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

To import the data, either copy the dump file to the replica, or access the file from the source when connecting remotely to the replica.

**17.1.2.5.2 Creating a Data Snapshot Using Raw Data Files**

This section describes how to create a data snapshot using the raw files which make up the database. Employing this method with a table using a storage engine that has complex caching or logging algorithms requires extra steps to produce a perfect “point in time” snapshot: the initial copy command could leave out cache information and logging updates, even if you have acquired a global read lock. How the storage engine responds to this depends on its crash recovery abilities.

If you use [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, you can use the **mysqlbackup** command from the MySQL Enterprise Backup component to produce a consistent snapshot. This command records the log name and offset corresponding to the snapshot to be used on the replica. MySQL Enterprise Backup is a commercial product that is included as part of a MySQL Enterprise subscription. See [Section 30.2, “MySQL Enterprise Backup Overview”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) for detailed information.

This method also does not work reliably if the source and replica have different values for [**ft\_stopword\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_stopword_file), [**ft\_min\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_min_word_len), or [**ft\_max\_word\_len**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ft_max_word_len) and you are copying tables having full-text indexes.

Assuming the above exceptions do not apply to your database, use the [cold backup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_cold_backup) technique to obtain a reliable binary snapshot of **InnoDB** tables: do a [slow shutdown](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_slow_shutdown) of the MySQL Server, then copy the data files manually.

To create a raw data snapshot of [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables when your MySQL data files exist on a single file system, you can use standard file copy tools such as **cp** or **copy**, a remote copy tool such as **scp** or **rsync**, an archiving tool such as **zip** or **tar**, or a file system snapshot tool such as **dump**. If you are replicating only certain databases, copy only those files that relate to those tables. For **InnoDB**, all tables in all databases are stored in the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) files, unless you have the [innodb\_file\_per\_table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) option enabled.

The following files are not required for replication:

Files relating to the **mysql** database.

The replica's connection metadata repository file **master.info**, if used; the use of this file is now deprecated (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)).

The source's binary log files, with the exception of the binary log index file if you are going to use this to locate the source binary log coordinates for the replica.

Any relay log files.

Depending on whether you are using **InnoDB** tables or not, choose one of the following:

If you are using [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, and also to get the most consistent results with a raw data snapshot, shut down the source server during the process, as follows:

Acquire a read lock and get the source's status. See [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus).

In a separate session, shut down the source server:

shell> **mysqladmin shutdown**

Make a copy of the MySQL data files. The following examples show common ways to do this. You need to choose only one of them:

shell> **tar cf */tmp/db.tar* *./data***

shell> **zip -r */tmp/db.zip* *./data***

shell> **rsync --recursive *./data* */tmp/dbdata***

Restart the source server.

If you are not using [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, you can get a snapshot of the system from a source without shutting down the server as described in the following steps:

Acquire a read lock and get the source's status. See [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus).

Make a copy of the MySQL data files. The following examples show common ways to do this. You need to choose only one of them:

shell> **tar cf */tmp/db.tar* *./data***

shell> **zip -r */tmp/db.zip* *./data***

shell> **rsync --recursive *./data* */tmp/dbdata***

In the client where you acquired the read lock, release the lock:

mysql> **UNLOCK TABLES;**

Once you have created the archive or copy of the database, copy the files to each replica before starting the replication process.

**17.1.2.6 Setting Up Replicas**

The following sections describe how to set up replicas. Before you proceed, ensure that you have:

Configured the source with the necessary configuration properties. See [Section 17.1.2.1, “Setting the Replication Source Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterbaseconfig).

Obtained the source status information, or a copy of the source's binary log index file made during a shutdown for the data snapshot. See [Section 17.1.2.4, “Obtaining the Replication Source Binary Log Coordinates”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterstatus).

On the source, released the read lock:

mysql> **UNLOCK TABLES;**

On the replica, edited the MySQL configuration. See [Section 17.1.2.2, “Setting the Replica Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slavebaseconfig).

The next steps depend on whether you have existing data to import to the replica or not. See [Section 17.1.2.5, “Choosing a Method for Data Snapshots”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-snapshot-method) for more information. Choose one of the following:

If you do not have a snapshot of a database to import, see [Section 17.1.2.6.1, “Setting Up Replication with New Source and Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-newservers).

If you have a snapshot of a database to import, see [Section 17.1.2.6.2, “Setting Up Replication with Existing Data”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-existingdata).

**17.1.2.6.1 Setting Up Replication with New Source and Replicas**

When there is no snapshot of a previous database to import, configure the replica to start replication from the new source.

To set up replication between a source and a new replica:

Start up the replica.

Execute a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement on the replica to set the source configuration. See [Section 17.1.2.7, “Setting the Source Configuration on the Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slaveinit).

Perform these replica setup steps on each replica.

This method can also be used if you are setting up new servers but have an existing dump of the databases from a different server that you want to load into your replication configuration. By loading the data into a new source, the data is automatically replicated to the replicas.

If you are setting up a new replication environment using the data from a different existing database server to create a new source, run the dump file generated from that server on the new source. The database updates are automatically propagated to the replicas:

shell> **mysql -h source < fulldb.dump**

**17.1.2.6.2 Setting Up Replication with Existing Data**

When setting up replication with existing data, transfer the snapshot from the source to the replica before starting replication. The process for importing data to the replica depends on how you created the snapshot of data on the source.

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

**Note**

If the replication source server or existing replica that you are copying to create the new replica has any scheduled events, ensure that these are disabled on the new replica before you start it. If an event runs on the new replica that has already run on the source, the duplicated operation causes an error. The Event Scheduler is controlled by the [**event\_scheduler**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_event_scheduler) system variable, which defaults to **ON** from MySQL 8.0, so events that are active on the original server run by default when the new replica starts up. To stop all events from running on the new replica, set the [**event\_scheduler**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_event_scheduler) system variable to **OFF** or **DISABLED** on the new replica. Alternatively, you can use the [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement to set individual events to **DISABLE** or **DISABLE ON SLAVE** to prevent them from running on the new replica. You can list the events on a server using the [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statement or the Information Schema [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table. For more information, see [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked).

As an alternative to creating a new replica in this way, MySQL Server's clone plugin can be used to transfer all the data and replication settings from an existing replica to a clone. For instructions to use this method, see [Section 5.6.7.6, “Cloning for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-replication).

Follow this procedure to set up replication with existing data:

If you used MySQL Server's clone plugin to create a clone from an existing replica (see [Section 5.6.7.6, “Cloning for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-replication)), the data is already transferred. Otherwise, import the data to the replica using one of the following methods.

If you used [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump), start the replica server, ensuring that replication does not start by using the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable. Then import the dump file:

shell> **mysql < fulldb.dump**

If you created a snapshot using the raw data files, extract the data files into your replica's data directory. For example:

shell> **tar xvf dbdump.tar**

You may need to set permissions and ownership on the files so that the replica server can access and modify them. Then start the replica server, ensuring that replication does not start by using the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable.

Configure the replica with the replication coordinates from the source. This tells the replica the binary log file and position within the file where replication needs to start. Also, configure the replica with the login credentials and host name of the source. For more information on the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement required, see [Section 17.1.2.7, “Setting the Source Configuration on the Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slaveinit).

Start the replication threads by issuing a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement.

After you have performed this procedure, the replica connects to the source and replicates any updates that have occurred on the source since the snapshot was taken. Error messages are issued to the replica's error log if it is not able to replicate for any reason.

The replica uses information logged in its connection metadata repository and applier metadata repository to keep track of how much of the source's binary log it has processed. From MySQL 8.0, by default, these repositories are tables named **slave\_master\_info** and **slave\_relay\_log\_info** in the **mysql** database. Do *not* remove or edit these tables unless you know exactly what you are doing and fully understand the implications. Even in that case, it is preferred that you use the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement to change replication parameters. The replica uses the values specified in the statement to update the replication metadata repositories automatically. See [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs), for more information.

**Note**

The contents of the replica's connection metadata repository override some of the server options specified on the command line or in my.cnf. See [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options), for more details.

A single snapshot of the source suffices for multiple replicas. To set up additional replicas, use the same source snapshot and follow the replica portion of the procedure just described.

**17.1.2.7 Setting the Source Configuration on the Replica**

To set up the replica to communicate with the source for replication, configure the replica with the necessary connection information. To do this, on the replica, execute the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23), replacing the option values with the actual values relevant to your system:

mysql> **CHANGE MASTER TO**

-> **MASTER\_HOST='*source\_host\_name*',**

-> **MASTER\_USER='*replication\_user\_name*',**

-> **MASTER\_PASSWORD='*replication\_password*',**

-> **MASTER\_LOG\_FILE='*recorded\_log\_file\_name*',**

-> **MASTER\_LOG\_POS=*recorded\_log\_position*;**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO**

-> **SOURCE\_HOST='*source\_host\_name*',**

-> **SOURCE\_USER='*replication\_user\_name*',**

-> **SOURCE\_PASSWORD='*replication\_password*',**

-> **SOURCE\_LOG\_FILE='*recorded\_log\_file\_name*',**

-> **SOURCE\_LOG\_POS=*recorded\_log\_position*;**

**Note**

Replication cannot use Unix socket files. You must be able to connect to the source MySQL server using TCP/IP.

The [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement has other options as well. For example, it is possible to set up secure replication using SSL. For a full list of options, and information about the maximum permissible length for the string-valued options, see [Section 13.4.2.1, “CHANGE MASTER TO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to).

**Important**

As noted in [Section 17.1.2.3, “Creating a User for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-repuser), if you are not using a secure connection and the user account named in the **SOURCE\_USER** | **MASTER\_USER** option authenticates with the **caching\_sha2\_password** plugin (the default from MySQL 8.0), you must specify the **SOURCE\_PUBLIC\_KEY\_PATH** | **MASTER\_PUBLIC\_KEY\_PATH** or **GET\_SOURCE\_PUBLIC\_KEY** | **GET\_MASTER\_PUBLIC\_KEY** option in the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | **CHANGE MASTER TO** statement to enable RSA key pair-based password exchange.

**17.1.2.8 Adding Replicas to a Replication Environment**

You can add another replica to an existing replication configuration without stopping the source server. To do this, you can set up the new replica by copying the data directory of an existing replica, and giving the new replica a different server ID (which is user-specified) and server UUID (which is generated at startup).

**Note**

If the replication source server or existing replica that you are copying to create the new replica has any scheduled events, ensure that these are disabled on the new replica before you start it. If an event runs on the new replica that has already run on the source, the duplicated operation causes an error. The Event Scheduler is controlled by the [**event\_scheduler**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_event_scheduler) system variable, which defaults to **ON** from MySQL 8.0, so events that are active on the original server run by default when the new replica starts up. To stop all events from running on the new replica, set the [**event\_scheduler**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_event_scheduler) system variable to **OFF** or **DISABLED** on the new replica. Alternatively, you can use the [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement to set individual events to **DISABLE** or **DISABLE ON SLAVE** to prevent them from running on the new replica. You can list the events on a server using the [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statement or the Information Schema [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table. For more information, see [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked).

As an alternative to creating a new replica in this way, MySQL Server's clone plugin can be used to transfer all the data and replication settings from an existing replica to a clone. For instructions to use this method, see [Section 5.6.7.6, “Cloning for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-replication).

To duplicate an existing replica without cloning, follow these steps:

Stop the existing replica and record the replica status information, particularly the source binary log file and relay log file positions. You can view the replica status either in the Performance Schema replication tables (see [Section 27.12.11, “Performance Schema Replication Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-tables)), or by issuing [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) as follows:

mysql> **STOP SLAVE;**

mysql> **SHOW SLAVE STATUS\G**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA;**

mysql> **SHOW REPLICA STATUS\G**

Shut down the existing replica:

shell> **mysqladmin shutdown**

Copy the data directory from the existing replica to the new replica, including the log files and relay log files. You can do this by creating an archive using **tar** or **WinZip**, or by performing a direct copy using a tool such as **cp** or **rsync**.

**Important**

Before copying, verify that all the files relating to the existing replica actually are stored in the data directory. For example, the **InnoDB** system tablespace, undo tablespace, and redo log might be stored in an alternative location. **InnoDB** tablespace files and file-per-table tablespaces might have been created in other directories. The binary logs and relay logs for the replica might be in their own directories outside the data directory. Check through the system variables that are set for the existing replica and look for any alternative paths that have been specified. If you find any, copy these directories over as well.

During copying, if files have been used for the replication metadata repositories (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)), ensure that you also copy these files from the existing replica to the new replica. If tables have been used for the repositories, which is the default from MySQL 8.0, the tables are in the data directory.

After copying, delete the auto.cnf file from the copy of the data directory on the new replica, so that the new replica is started with a different generated server UUID. The server UUID must be unique.

A common problem that is encountered when adding new replicas is that the new replica fails with a series of warning and error messages like these:

071118 16:44:10 [Warning] Neither --relay-log nor --relay-log-index were used; so

replication may break when this MySQL server acts as a replica and has his hostname

changed!! Please use '--relay-log=***new\_replica\_hostname***-relay-bin' to avoid this problem.

071118 16:44:10 [ERROR] Failed to open the relay log './***old\_replica\_hostname***-relay-bin.003525'

(relay\_log\_pos 22940879)

071118 16:44:10 [ERROR] Could not find target log during relay log initialization

071118 16:44:10 [ERROR] Failed to initialize the master info structure

This situation can occur if the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable is not specified, as the relay log files contain the host name as part of their file names. This is also true of the relay log index file if the [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variable is not used. For more information about these variables, see [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options).

To avoid this problem, use the same value for [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) on the new replica that was used on the existing replica. If this option was not set explicitly on the existing replica, use ***existing\_replica\_hostname*-relay-bin**. If this is not possible, copy the existing replica's relay log index file to the new replica and set the [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variable on the new replica to match what was used on the existing replica. If this option was not set explicitly on the existing replica, use ***existing\_replica\_hostname*-relay-bin.index**. Alternatively, if you have already tried to start the new replica after following the remaining steps in this section and have encountered errors like those described previously, then perform the following steps:

If you have not already done so, issue [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) on the new replica.

If you have already started the existing replica again, issue [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) on the existing replica as well.

Copy the contents of the existing replica's relay log index file into the new replica's relay log index file, making sure to overwrite any content already in the file.

Proceed with the remaining steps in this section.

When copying is complete, restart the existing replica.

On the new replica, edit the configuration and give the new replica a unique server ID (using the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable) that is not used by the source or any of the existing replicas.

Start the new replica server, ensuring that replication does not start yet by specifying the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable. Use the Performance Schema replication tables or issue [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) to confirm that the new replica has the correct settings when compared with the existing replica. Also display the server ID and server UUID and verify that these are correct and unique for the new replica.

Start the replica threads by issuing a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement. The new replica now uses the information in its connection metadata repository to start the replication process.

**17.1.3 Replication with Global Transaction Identifiers**

[17.1.3.1 GTID Format and Storage](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts)

[17.1.3.2 GTID Life Cycle](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-lifecycle)

[17.1.3.3 GTID Auto-Positioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning)

[17.1.3.4 Setting Up Replication Using GTIDs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto)

[17.1.3.5 Using GTIDs for Failover and Scaleout](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover)

[17.1.3.6 Replication From a Source Without GTIDs to a Replica With GTIDs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-assign-anon)

[17.1.3.7 Restrictions on Replication with GTIDs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restrictions)

[17.1.3.8 Stored Function Examples to Manipulate GTIDs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-functions)

This section explains transaction-based replication using global transaction identifiers (GTIDs). When using GTIDs, each transaction can be identified and tracked as it is committed on the originating server and applied by any replicas; this means that it is not necessary when using GTIDs to refer to log files or positions within those files when starting a new replica or failing over to a new source, which greatly simplifies these tasks. Because GTID-based replication is completely transaction-based, it is simple to determine whether sources and replicas are consistent; as long as all transactions committed on a source are also committed on a replica, consistency between the two is guaranteed. You can use either statement-based or row-based replication with GTIDs (see [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats)); however, for best results, we recommend that you use the row-based format.

GTIDs are always preserved between source and replica. This means that you can always determine the source for any transaction applied on any replica by examining its binary log. In addition, once a transaction with a given GTID is committed on a given server, any subsequent transaction having the same GTID is ignored by that server. Thus, a transaction committed on the source can be applied no more than once on the replica, which helps to guarantee consistency.

This section discusses the following topics:

How GTIDs are defined and created, and how they are represented in a MySQL server (see [Section 17.1.3.1, “GTID Format and Storage”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts)).

The life cycle of a GTID (see [Section 17.1.3.2, “GTID Life Cycle”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-lifecycle)).

The auto-positioning function for synchronizing a replica and source that use GTIDs (see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning)).

A general procedure for setting up and starting GTID-based replication (see [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto)).

Suggested methods for provisioning new replication servers when using GTIDs (see [Section 17.1.3.5, “Using GTIDs for Failover and Scaleout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover)).

Restrictions and limitations that you should be aware of when using GTID-based replication (see [Section 17.1.3.7, “Restrictions on Replication with GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restrictions)).

Stored functions that you can use to work with GTIDs (see [Section 17.1.3.8, “Stored Function Examples to Manipulate GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-functions)).

For information about MySQL Server options and variables relating to GTID-based replication, see [Section 17.1.6.5, “Global Transaction ID System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-gtids). See also [Section 12.19, “Functions Used with Global Transaction Identifiers (GTIDs)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gtid-functions), which describes SQL functions supported by MySQL 8.0 for use with GTIDs.

**17.1.3.1 GTID Format and Storage**

A global transaction identifier (GTID) is a unique identifier created and associated with each transaction committed on the server of origin (the source). This identifier is unique not only to the server on which it originated, but is unique across all servers in a given replication topology.

GTID assignment distinguishes between client transactions, which are committed on the source, and replicated transactions, which are reproduced on a replica. When a client transaction is committed on the source, it is assigned a new GTID, provided that the transaction was written to the binary log. Client transactions are guaranteed to have monotonically increasing GTIDs without gaps between the generated numbers. If a client transaction is not written to the binary log (for example, because the transaction was filtered out, or the transaction was read-only), it is not assigned a GTID on the server of origin.

Replicated transactions retain the same GTID that was assigned to the transaction on the server of origin. The GTID is present before the replicated transaction begins to execute, and is persisted even if the replicated transaction is not written to the binary log on the replica, or is filtered out on the replica. The MySQL system table **mysql.gtid\_executed** is used to preserve the assigned GTIDs of all the transactions applied on a MySQL server, except those that are stored in a currently active binary log file.

The auto-skip function for GTIDs means that a transaction committed on the source can be applied no more than once on the replica, which helps to guarantee consistency. Once a transaction with a given GTID has been committed on a given server, any attempt to execute a subsequent transaction with the same GTID is ignored by that server. No error is raised, and no statement in the transaction is executed.

If a transaction with a given GTID has started to execute on a server, but has not yet committed or rolled back, any attempt to start a concurrent transaction on the server with the same GTID blocks. The server neither begins to execute the concurrent transaction nor returns control to the client. Once the first attempt at the transaction commits or rolls back, concurrent sessions that were blocking on the same GTID may proceed. If the first attempt rolled back, one concurrent session proceeds to attempt the transaction, and any other concurrent sessions that were blocking on the same GTID remain blocked. If the first attempt committed, all the concurrent sessions stop being blocked, and auto-skip all the statements of the transaction.

A GTID is represented as a pair of coordinates, separated by a colon character (**:**), as shown here:

GTID = ***source\_id***:***transaction\_id***

The ***source\_id*** identifies the originating server. Normally, the source's [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) is used for this purpose. The ***transaction\_id*** is a sequence number determined by the order in which the transaction was committed on the source. For example, the first transaction to be committed has **1** as its ***transaction\_id***, and the tenth transaction to be committed on the same originating server is assigned a ***transaction\_id*** of **10**. It is not possible for a transaction to have **0** as a sequence number in a GTID. For example, the twenty-third transaction to be committed originally on the server with the UUID **3E11FA47-71CA-11E1-9E33-C80AA9429562** has this GTID:

3E11FA47-71CA-11E1-9E33-C80AA9429562:23

The upper limit for sequence numbers for GTIDs on a server instance is the number of non-negative values for a signed 64-bit integer (2 to the power of 63 minus 1, or 9,223,372,036,854,775,807). If the server runs out of GTIDs, it takes the action specified by [**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action). From MySQL 8.0.23, a warning message is issued when the server instance is approaching the limit.

The GTID for a transaction is shown in the output from [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), and it is used to identify an individual transaction in the Performance Schema replication status tables, for example, [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table). The value stored by the [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) system variable (**@@GLOBAL.gtid\_next**) is a single GTID.

**GTID Sets**

A GTID set is a set comprising one or more single GTIDs or ranges of GTIDs. GTID sets are used in a MySQL server in several ways. For example, the values stored by the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variables are GTID sets. The [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) clauses **UNTIL SQL\_BEFORE\_GTIDS** and **UNTIL SQL\_AFTER\_GTIDS** can be used to make a replica process transactions only up to the first GTID in a GTID set, or stop after the last GTID in a GTID set. The built-in functions [**GTID\_SUBSET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subset) and [**GTID\_SUBTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subtract) require GTID sets as input.

A range of GTIDs originating from the same server can be collapsed into a single expression, as shown here:

3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5

The above example represents the first through fifth transactions originating on the MySQL server whose [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) is **3E11FA47-71CA-11E1-9E33-C80AA9429562**. Multiple single GTIDs or ranges of GTIDs originating from the same server can also be included in a single expression, with the GTIDs or ranges separated by colons, as in the following example:

3E11FA47-71CA-11E1-9E33-C80AA9429562:1-3:11:47-49

A GTID set can include any combination of single GTIDs and ranges of GTIDs, and it can include GTIDs originating from different servers. This example shows the GTID set stored in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable (**@@GLOBAL.gtid\_executed**) of a replica that has applied transactions from more than one source:

2174B383-5441-11E8-B90A-C80AA9429562:1-3, 24DA167-0C0C-11E8-8442-00059A3C7B00:1-19

When GTID sets are returned from server variables, UUIDs are in alphabetical order, and numeric intervals are merged and in ascending order.

The syntax for a GTID set is as follows:

***gtid\_set***:

***uuid\_set*** [, ***uuid\_set***] ...

| ''

***uuid\_set***:

***uuid***:***interval***[:***interval***]...

***uuid***:

***hhhhhhhh***-***hhhh***-***hhhh***-***hhhh***-***hhhhhhhhhhhh***

***h***:

[0-9|A-F]

***interval***:

***n***[-***n***]

(***n*** >= 1)

**mysql.gtid\_executed Table**

GTIDs are stored in a table named **gtid\_executed**, in the **mysql** database. A row in this table contains, for each GTID or set of GTIDs that it represents, the UUID of the originating server, and the starting and ending transaction IDs of the set; for a row referencing only a single GTID, these last two values are the same.

The **mysql.gtid\_executed** table is created (if it does not already exist) when MySQL Server is installed or upgraded, using a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement similar to that shown here:

CREATE TABLE gtid\_executed (

source\_uuid CHAR(36) NOT NULL,

interval\_start BIGINT(20) NOT NULL,

interval\_end BIGINT(20) NOT NULL,

PRIMARY KEY (source\_uuid, interval\_start)

)

**Warning**

As with other MySQL system tables, do not attempt to create or modify this table yourself.

The **mysql.gtid\_executed** table is provided for internal use by the MySQL server. It enables a replica to use GTIDs when binary logging is disabled on the replica, and it enables retention of the GTID state when the binary logs have been lost. Note that the **mysql.gtid\_executed** table is cleared if you issue [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master).

GTIDs are stored in the **mysql.gtid\_executed** table only when [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **ON** or **ON\_PERMISSIVE**. If binary logging is disabled (**log\_bin** is **OFF**), or if [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) is disabled, the server stores the GTID belonging to each transaction together with the transaction in the **mysql.gtid\_executed** table at transaction commit time. In addition, the table is compressed periodically at a user-configurable rate, as described in [mysql.gtid\_executed Table Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-executed-table-compression).

If binary logging is enabled (**log\_bin** is **ON**), from MySQL 8.0.17 for the **InnoDB** storage engine only, the server updates the **mysql.gtid\_executed** table in the same way as when binary logging or replica update logging is disabled, storing the GTID for each transaction at transaction commit time. However, in releases before MySQL 8.0.17, and for other storage engines, the server only updates the **mysql.gtid\_executed** table when the binary log is rotated or the server is shut down. At these times, the server writes GTIDs for all transactions that were written into the previous binary log into the **mysql.gtid\_executed** table. This situation applies on a source prior to MySQL 8.0.17, or on a replica prior to MySQL 8.0.17 where binary logging is enabled, or with storage engines other than **InnoDB**, it has the following consequences:

In the event of the server stopping unexpectedly, the set of GTIDs from the current binary log file is not saved in the **mysql.gtid\_executed** table. These GTIDs are added to the table from the binary log file during recovery so that replication can continue. The exception to this is if you disable binary logging when the server is restarted (using [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) or [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin)). In that case, the server cannot access the binary log file to recover the GTIDs, so replication cannot be started.

The **mysql.gtid\_executed** table does not hold a complete record of the GTIDs for all executed transactions. That information is provided by the global value of the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable. In releases before MySQL 8.0.17 and with storage engines other than **InnoDB**, always use **@@GLOBAL.gtid\_executed**, which is updated after every commit, to represent the GTID state for the MySQL server, instead of querying the **mysql.gtid\_executed** table.

The MySQL server can write to the **mysql.gtid\_executed** table even when the server is in read only or super read only mode. In releases before MySQL 8.0.17, this ensures that the binary log file can still be rotated in these modes. If the **mysql.gtid\_executed** table cannot be accessed for writes, and the binary log file is rotated for any reason other than reaching the maximum file size ([**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size)), the current binary log file continues to be used. An error message is returned to the client that requested the rotation, and a warning is logged on the server. If the **mysql.gtid\_executed** table cannot be accessed for writes and [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) is reached, the server responds according to its [**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action) setting. If **IGNORE\_ERROR** is set, an error is logged on the server and binary logging is halted, or if **ABORT\_SERVER** is set, the server shuts down.

**mysql.gtid\_executed Table Compression**

Over the course of time, the **mysql.gtid\_executed** table can become filled with many rows referring to individual GTIDs that originate on the same server, and whose transaction IDs make up a range, similar to what is shown here:

+--------------------------------------+----------------+--------------+

| source\_uuid | interval\_start | interval\_end |

|--------------------------------------+----------------+--------------|

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 37 | 37 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 38 | 38 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 39 | 39 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 40 | 40 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 41 | 41 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 42 | 42 |

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 43 | 43 |

...

To save space, the MySQL server can compress the **mysql.gtid\_executed** table periodically by replacing each such set of rows with a single row that spans the entire interval of transaction identifiers, like this:

+--------------------------------------+----------------+--------------+

| source\_uuid | interval\_start | interval\_end |

|--------------------------------------+----------------+--------------|

| 3E11FA47-71CA-11E1-9E33-C80AA9429562 | 37 | 43 |

...

The server can carry out compression using a dedicated foreground thread named **thread/sql/compress\_gtid\_table**. This thread is not listed in the output of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist), but it can be viewed as a row in the [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table, as shown here:

mysql> **SELECT \* FROM performance\_schema.threads WHERE NAME LIKE '%gtid%'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THREAD\_ID: 26

NAME: thread/sql/compress\_gtid\_table

TYPE: FOREGROUND

PROCESSLIST\_ID: 1

PROCESSLIST\_USER: NULL

PROCESSLIST\_HOST: NULL

PROCESSLIST\_DB: NULL

PROCESSLIST\_COMMAND: Daemon

PROCESSLIST\_TIME: 1509

PROCESSLIST\_STATE: Suspending

PROCESSLIST\_INFO: NULL

PARENT\_THREAD\_ID: 1

ROLE: NULL

INSTRUMENTED: YES

HISTORY: YES

CONNECTION\_TYPE: NULL

THREAD\_OS\_ID: 18677

When binary logging is enabled on the server, this compression method is not used, and instead the **mysql.gtid\_executed** table is compressed on each binary log rotation. However, when binary logging is disabled on the server, the **thread/sql/compress\_gtid\_table** thread sleeps until a specified number of transactions have been executed, then wakes up to perform compression of the **mysql.gtid\_executed** table. It then sleeps until the same number of transactions have taken place, then wakes up to perform the compression again, repeating this loop indefinitely. The number of transactions that elapse before the table is compressed, and thus the compression rate, is controlled by the value of the [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) system variable. Setting that value to 0 means that the thread never wakes up, meaning that this explicit compression method is not used. Instead, compression occurs implicitly as required.

From MySQL 8.0.17, [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) transactions are written to the **mysql.gtid\_executed** table by a separate process to non-**InnoDB** transactions. This process is controlled by a different thread, **innodb/clone\_gtid\_thread**. This GTID persister thread collects GTIDs in groups, flushes them to the **mysql.gtid\_executed** table, then compresses the table. If the server has a mix of [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) transactions and non-[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) transactions, which are written to the **mysql.gtid\_executed** table individually, the compression carried out by the **compress\_gtid\_table** thread interferes with the work of the GTID persister thread and can slow it significantly. For this reason, from that release it is recommended that you set [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) to 0, so that the **compress\_gtid\_table** thread is never activated.

From MySQL 8.0.23, the [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) default value is 0, and both **InnoDB** and non-**InnoDB** transactions are written to the **mysql.gtid\_executed** table by the GTID persister thread.

For releases before MySQL 8.0.17, the default value of 1000 for [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) can be used, meaning that compression of the table is performed after each 1000 transactions, or you can choose an alternative value. In those releases, if you set a value of 0 and binary logging is disabled, explicit compression is not performed on the **mysql.gtid\_executed** table, and you should be prepared for a potentially large increase in the amount of disk space that may be required by the table if you do this.

When a server instance is started, if [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) is set to a nonzero value and the **thread/sql/compress\_gtid\_table** thread is launched, in most server configurations, explicit compression is performed for the **mysql.gtid\_executed** table. In releases before MySQL 8.0.17 when binary logging is enabled, compression is triggered by the fact of the binary log being rotated at startup. In releases from MySQL 8.0.20, compression is triggered by the thread launch. In the intervening releases, compression does not take place at startup.

**17.1.3.2 GTID Life Cycle**

The life cycle of a GTID consists of the following steps:

A transaction is executed and committed on the source. This client transaction is assigned a GTID composed of the source's UUID and the smallest nonzero transaction sequence number not yet used on this server. The GTID is written to the source's binary log (immediately preceding the transaction itself in the log). If a client transaction is not written to the binary log (for example, because the transaction was filtered out, or the transaction was read-only), it is not assigned a GTID.

If a GTID was assigned for the transaction, the GTID is persisted atomically at commit time by writing it to the binary log at the beginning of the transaction (as a **Gtid\_log\_event**). Whenever the binary log is rotated or the server is shut down, the server writes GTIDs for all transactions that were written into the previous binary log file into the **mysql.gtid\_executed** table.

If a GTID was assigned for the transaction, the GTID is externalized non-atomically (very shortly after the transaction is committed) by adding it to the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable (**@@GLOBAL.gtid\_executed**). This GTID set contains a representation of the set of all committed GTID transactions, and it is used in replication as a token that represents the server state. With binary logging enabled (as required for the source), the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable is a complete record of the transactions applied, but the **mysql.gtid\_executed** table is not, because the most recent history is still in the current binary log file.

After the binary log data is transmitted to the replica and stored in the replica's relay log (using established mechanisms for this process, see [Section 17.2, “Replication Implementation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation), for details), the replica reads the GTID and sets the value of its [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) system variable as this GTID. This tells the replica that the next transaction must be logged using this GTID. It is important to note that the replica sets **gtid\_next** in a session context.

The replica verifies that no thread has yet taken ownership of the GTID in [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) in order to process the transaction. By reading and checking the replicated transaction's GTID first, before processing the transaction itself, the replica guarantees not only that no previous transaction having this GTID has been applied on the replica, but also that no other session has already read this GTID but has not yet committed the associated transaction. So if multiple clients attempt to apply the same transaction concurrently, the server resolves this by letting only one of them execute. The [**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned) system variable (**@@GLOBAL.gtid\_owned**) for the replica shows each GTID that is currently in use and the ID of the thread that owns it. If the GTID has already been used, no error is raised, and the auto-skip function is used to ignore the transaction.

If the GTID has not been used, the replica applies the replicated transaction. Because [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) is set to the GTID already assigned by the source, the replica does not attempt to generate a new GTID for this transaction, but instead uses the GTID stored in [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next).

If binary logging is enabled on the replica, the GTID is persisted atomically at commit time by writing it to the binary log at the beginning of the transaction (as a **Gtid\_log\_event**). Whenever the binary log is rotated or the server is shut down, the server writes GTIDs for all transactions that were written into the previous binary log file into the **mysql.gtid\_executed** table.

If binary logging is disabled on the replica, the GTID is persisted atomically by writing it directly into the **mysql.gtid\_executed** table. MySQL appends a statement to the transaction to insert the GTID into the table. From MySQL 8.0, this operation is atomic for DDL statements as well as for DML statements. In this situation, the **mysql.gtid\_executed** table is a complete record of the transactions applied on the replica.

Very shortly after the replicated transaction is committed on the replica, the GTID is externalized non-atomically by adding it to the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable (**@@GLOBAL.gtid\_executed**) for the replica. As for the source, this GTID set contains a representation of the set of all committed GTID transactions. If binary logging is disabled on the replica, the **mysql.gtid\_executed** table is also a complete record of the transactions applied on the replica. If binary logging is enabled on the replica, meaning that some GTIDs are only recorded in the binary log, the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable is the only complete record.

Client transactions that are completely filtered out on the source are not assigned a GTID, therefore they are not added to the set of transactions in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable, or added to the **mysql.gtid\_executed** table. However, the GTIDs of replicated transactions that are completely filtered out on the replica are persisted. If binary logging is enabled on the replica, the filtered-out transaction is written to the binary log as a **Gtid\_log\_event** followed by an empty transaction containing only **BEGIN** and **COMMIT** statements. If binary logging is disabled, the GTID of the filtered-out transaction is written to the **mysql.gtid\_executed** table. Preserving the GTIDs for filtered-out transactions ensures that the **mysql.gtid\_executed** table and the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable can be compressed. It also ensures that the filtered-out transactions are not retrieved again if the replica reconnects to the source, as explained in [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning).

On a multithreaded replica (with [**slave\_parallel\_workers > 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) ), transactions can be applied in parallel, so replicated transactions can commit out of order (unless [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) is set). When that happens, the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable contains multiple GTID ranges with gaps between them. (On a source or a single-threaded replica, there are monotonically increasing GTIDs without gaps between the numbers.) Gaps on multithreaded replicas only occur among the most recently applied transactions, and are filled in as replication progresses. When replication threads are stopped cleanly using the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement, ongoing transactions are applied so that the gaps are filled in. In the event of a shutdown such as a server failure or the use of the [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) statement to stop replication threads, the gaps might remain.

**What changes are assigned a GTID?**

The typical scenario is that the server generates a new GTID for a committed transaction. However, GTIDs can also be assigned to other changes besides transactions, and in some cases a single transaction can be assigned multiple GTIDs.

Every database change (DDL or DML) that is written to the binary log is assigned a GTID. This includes changes that are autocommitted, and changes that are committed using **BEGIN** and **COMMIT** or **START TRANSACTION** statements. A GTID is also assigned to the creation, alteration, or deletion of a database, and of a non-table database object such as a procedure, function, trigger, event, view, user, role, or grant.

Non-transactional updates as well as transactional updates are assigned GTIDs. In addition, for a non-transactional update, if a disk write failure occurs while attempting to write to the binary log cache and a gap is therefore created in the binary log, the resulting incident log event is assigned a GTID.

When a table is automatically dropped by a generated statement in the binary log, a GTID is assigned to the statement. Temporary tables are dropped automatically when a replica begins to apply events from a source that has just been started, and when statement-based replication is in use ([**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)) and a user session that has open temporary tables disconnects. Tables that use the [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) storage engine are deleted automatically the first time they are accessed after the server is started, because rows might have been lost during the shutdown.

When a transaction is not written to the binary log on the server of origin, the server does not assign a GTID to it. This includes transactions that are rolled back and transactions that are executed while binary logging is disabled on the server of origin, either globally (with **--skip-log-bin** specified in the server's configuration) or for the session (**SET @@SESSION.sql\_log\_bin = 0**). This also includes no-op transactions when row-based replication is in use ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)).

XA transactions are assigned separate GTIDs for the **XA PREPARE** phase of the transaction and the **XA COMMIT** or **XA ROLLBACK** phase of the transaction. XA transactions are persistently prepared so that users can commit them or roll them back in the case of a failure (which in a replication topology might include a failover to another server). The two parts of the transaction are therefore replicated separately, so they must have their own GTIDs, even though a non-XA transaction that is rolled back would not have a GTID.

In the following special cases, a single statement can generate multiple transactions, and therefore be assigned multiple GTIDs:

A stored procedure is invoked that commits multiple transactions. One GTID is generated for each transaction that the procedure commits.

A multi-table [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement drops tables of different types. Multiple GTIDs can be generated if any of the tables use storage engines that do not support atomic DDL, or if any of the tables are temporary tables.

A [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement is issued when row-based replication is in use ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)). One GTID is generated for the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) action and one GTID is generated for the row-insert actions.

**The gtid\_next System Variable**

By default, for new transactions committed in user sessions, the server automatically generates and assigns a new GTID. When the transaction is applied on a replica, the GTID from the server of origin is preserved. You can change this behavior by setting the session value of the [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) system variable:

When [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) is set to **AUTOMATIC**, which is the default, and a transaction is committed and written to the binary log, the server automatically generates and assigns a new GTID. If a transaction is rolled back or not written to the binary log for another reason, the server does not generate and assign a GTID.

If you set [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) to a valid GTID (consisting of a UUID and a transaction sequence number, separated by a colon), the server assigns that GTID to your transaction. This GTID is assigned and added to [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) even when the transaction is not written to the binary log, or when the transaction is empty.

Note that after you set [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) to a specific GTID, and the transaction has been committed or rolled back, an explicit **SET @@SESSION.gtid\_next** statement must be issued before any other statement. You can use this to set the GTID value back to **AUTOMATIC** if you do not want to assign any more GTIDs explicitly.

When replication applier threads apply replicated transactions, they use this technique, setting **@@SESSION.gtid\_next** explicitly to the GTID of the replicated transaction as assigned on the server of origin. This means the GTID from the server of origin is retained, rather than a new GTID being generated and assigned by the replica. It also means the GTID is added to [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) on the replica even when binary logging or replica update logging is disabled on the replica, or when the transaction is a no-op or is filtered out on the replica.

It is possible for a client to simulate a replicated transaction by setting **@@SESSION.gtid\_next** to a specific GTID before executing the transaction. This technique is used by [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to generate a dump of the binary log that the client can replay to preserve GTIDs. A simulated replicated transaction committed through a client is completely equivalent to a replicated transaction committed through a replication applier thread, and they cannot be distinguished after the fact.

**The gtid\_purged System Variable**

The set of GTIDs in the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable (**@@GLOBAL.gtid\_purged**) contains the GTIDs of all the transactions that have been committed on the server, but do not exist in any binary log file on the server. [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) is a subset of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). The following categories of GTIDs are in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged):

GTIDs of replicated transactions that were committed with binary logging disabled on the replica.

GTIDs of transactions that were written to a binary log file that has now been purged.

GTIDs that were added explicitly to the set by the statement **SET @@GLOBAL.gtid\_purged**.

You can change the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) in order to record on the server that the transactions in a certain GTID set have been applied, although they do not exist in any binary log on the server. When you add GTIDs to [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), they are also added to [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). An example use case for this action is when you are restoring a backup of one or more databases on a server, but you do not have the relevant binary logs containing the transactions on the server. Before MySQL 8.0, you could only change the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) when [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) (and therefore [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged)) was empty. From MySQL 8.0, this restriction does not apply, and you can also choose whether to replace the whole GTID set in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) with a specified GTID set, or to add a specified GTID set to the GTIDs already in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged). For details of how to do this, see the description for [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged).

The sets of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variables are initialized when the server starts. Every binary log file begins with the event **Previous\_gtids\_log\_event**, which contains the set of GTIDs in all previous binary log files (composed from the GTIDs in the preceding file's **Previous\_gtids\_log\_event**, and the GTIDs of every **Gtid\_log\_event** in the preceding file itself). The contents of **Previous\_gtids\_log\_event** in the oldest and most recent binary log files are used to compute the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) sets at server startup:

[**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) is computed as the union of the GTIDs in **Previous\_gtids\_log\_event** in the most recent binary log file, the GTIDs of transactions in that binary log file, and the GTIDs stored in the **mysql.gtid\_executed** table. This GTID set contains all the GTIDs that have been used (or added explicitly to [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged)) on the server, whether or not they are currently in a binary log file on the server. It does not include the GTIDs for transactions that are currently being processed on the server (**@@GLOBAL.gtid\_owned**).

[**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) is computed by first adding the GTIDs in **Previous\_gtids\_log\_event** in the most recent binary log file and the GTIDs of transactions in that binary log file. This step gives the set of GTIDs that are currently, or were once, recorded in a binary log on the server (**gtids\_in\_binlog**). Next, the GTIDs in **Previous\_gtids\_log\_event** in the oldest binary log file are subtracted from **gtids\_in\_binlog**. This step gives the set of GTIDs that are currently recorded in a binary log on the server (**gtids\_in\_binlog\_not\_purged**). Finally, **gtids\_in\_binlog\_not\_purged** is subtracted from [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). The result is the set of GTIDs that have been used on the server, but are not currently recorded in a binary log file on the server, and this result is used to initialize [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged).

If binary logs from MySQL 5.7.7 or older are involved in these computations, it is possible for incorrect GTID sets to be computed for [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), and they remain incorrect even if the server is later restarted. For details, see the description for the [**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) system variable, which controls how the binary logs are iterated to compute the GTID sets. If one of the situations described there applies on a server, set [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) in the server's configuration file before starting it. That setting makes the server iterate all the binary log files (not just the newest and oldest) to find where GTID events start to appear. This process could take a long time if the server has a large number of binary log files without GTID events.

**Resetting the GTID Execution History**

If you need to reset the GTID execution history on a server, use the [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) statement. For example, you might need to do this after carrying out test queries to verify a replication setup on new GTID-enabled servers, or when you want to join a new server to a replication group but it contains some unwanted local transactions that are not accepted by Group Replication.

**Warning**

Use [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) with caution to avoid losing any wanted GTID execution history and binary log files.

Before issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master), ensure that you have backups of the server's binary log files and binary log index file, if any, and obtain and save the GTID set held in the global value of the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable (for example, by issuing a **SELECT @@GLOBAL.gtid\_executed** statement and saving the results). If you are removing unwanted transactions from that GTID set, use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to examine the contents of the transactions to ensure that they have no value, contain no data that must be saved or replicated, and did not result in data changes on the server.

When you issue [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master), the following reset operations are carried out:

The value of the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable is set to an empty string (**''**).

The global value (but not the session value) of the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable is set to an empty string.

The **mysql.gtid\_executed** table is cleared (see [mysql.gtid\_executed Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-executed-table)).

If the server has binary logging enabled, the existing binary log files are deleted and the binary log index file is cleared.

Note that [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) is the method to reset the GTID execution history even if the server is a replica where binary logging is disabled. [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) has no effect on the GTID execution history.

**17.1.3.3 GTID Auto-Positioning**

GTIDs replace the file-offset pairs previously required to determine points for starting, stopping, or resuming the flow of data between source and replica. When GTIDs are in use, all the information that the replica needs for synchronizing with the source is obtained directly from the replication data stream.

To start a replica using GTID-based replication, you need to enable the **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option in the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). The alternative **SOURCE\_LOG\_FILE** | **MASTER\_LOG\_FILE** and **SOURCE\_LOG\_POS** | **MASTER\_LOG\_POS** options specify the name of the log file and the starting position within the file, but with GTIDs the replica does not need this nonlocal data.. For full instructions to configure and start sources and replicas using GTID-based replication, see [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto).

The **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option is disabled by default. If multi-source replication is enabled on the replica, you need to set the option for each applicable replication channel. Disabling the **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option again makes the replica revert to file-based replication, in which case you must also specify one or both of the **SOURCE\_LOG\_FILE** | **MASTER\_LOG\_FILE** or **SOURCE\_LOG\_POS** | **MASTER\_LOG\_POS** options.

When a replica has GTIDs enabled ([**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), **ON\_PERMISSIVE,** or **OFF\_PERMISSIVE** ) and the **MASTER\_AUTO\_POSITION** option enabled, auto-positioning is activated for connection to the source. The source must have [**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set in order for the connection to succeed. In the initial handshake, the replica sends a GTID set containing the transactions that it has already received, committed, or both. This GTID set is equal to the union of the set of GTIDs in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable (**@@GLOBAL.gtid\_executed**), and the set of GTIDs recorded in the Performance Schema [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) table as received transactions (the result of the statement **SELECT RECEIVED\_TRANSACTION\_SET FROM PERFORMANCE\_SCHEMA.replication\_connection\_status**).

The source responds by sending all transactions recorded in its binary log whose GTID is not included in the GTID set sent by the replica. To do this, the source first identifies the appropriate binary log file to begin working with, by checking the **Previous\_gtids\_log\_event** in the header of each of its binary log files, starting with the most recent. When the source finds the first **Previous\_gtids\_log\_event** which contains no transactions that the replica is missing, it begins with that binary log file. This method is efficient and only takes a significant amount of time if the replica is behind the source by a large number of binary log files. The source then reads the transactions in that binary log file and subsequent files up to the current one, sending the transactions with GTIDs that the replica is missing, and skipping the transactions that were in the GTID set sent by the replica. The elapsed time until the replica receives the first missing transaction depends on its offset in the binary log file. This exchange ensures that the source only sends the transactions with a GTID that the replica has not already received or committed. If the replica receives transactions from more than one source, as in the case of a diamond topology, the auto-skip function ensures that the transactions are not applied twice.

If any of the transactions that should be sent by the source have been purged from the source's binary log, or added to the set of GTIDs in the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable by another method, the source sends the error **ER\_MASTER\_HAS\_PURGED\_REQUIRED\_GTIDS** to the replica, and replication does not start. The GTIDs of the missing purged transactions are identified and listed in the source's error log in the warning message **ER\_FOUND\_MISSING\_GTIDS**. The replica cannot recover automatically from this error because parts of the transaction history that are needed to catch up with the source have been purged. Attempting to reconnect without the **MASTER\_AUTO\_POSITION** option enabled only results in the loss of the purged transactions on the replica. The correct approach to recover from this situation is for the replica to replicate the missing transactions listed in the **ER\_FOUND\_MISSING\_GTIDS** message from another source, or for the replica to be replaced by a new replica created from a more recent backup. Consider revising the binary log expiration period ([**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds)) on the source to ensure that the situation does not occur again.

If during the exchange of transactions it is found that the replica has received or committed transactions with the source's UUID in the GTID, but the source itself does not have a record of them, the source sends the error **ER\_SLAVE\_HAS\_MORE\_GTIDS\_THAN\_MASTER** to the replica and replication does not start. This situation can occur if a source that does not have [**sync\_binlog=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) set experiences a power failure or operating system crash, and loses committed transactions that have not yet been synchronized to the binary log file, but have been received by the replica. The source and replica can diverge if any clients commit transactions on the source after it is restarted, which can lead to the situation where the source and replica are using the same GTID for different transactions. The correct approach to recover from this situation is to check manually whether the source and replica have diverged. If the same GTID is now in use for different transactions, you either need to perform manual conflict resolution for individual transactions as required, or remove either the source or the replica from the replication topology. If the issue is only missing transactions on the source, you can make the source into a replica instead, allow it to catch up with the other servers in the replication topology, and then make it a source again if needed.

For a multi-source replica in a diamond topology (where the replica replicates from two or more sources, which in turn replicate from a common source), when GTID-based replication is in use, ensure that any replication filters or other channel configuration are identical on all channels on the multi-source replica. With GTID-based replication, filters are applied only to the transaction data, and GTIDs are not filtered out. This happens so that a replica’s GTID set stays consistent with the source’s, meaning GTID auto-positioning can be used without re-acquiring filtered out transactions each time. In the case where the downstream replica is multi-source and receives the same transaction from multiple sources in a diamond topology, the downstream replica now has multiple versions of the transaction, and the result depends on which channel applies the transaction first. The second channel to attempt it skips the transaction using GTID auto-skip, because the transaction’s GTID was added to the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set by the first channel. With identical filtering on the channels, there is no problem because all versions of the transaction contain the same data, so the results are the same. However, with different filtering on the channels, the database can become inconsistent and replication can hang.

**17.1.3.4 Setting Up Replication Using GTIDs**

This section describes a process for configuring and starting GTID-based replication in MySQL 8.0. This is a “cold start” procedure that assumes either that you are starting the source server for the first time, or that it is possible to stop it; for information about provisioning replicas using GTIDs from a running source server, see [Section 17.1.3.5, “Using GTIDs for Failover and Scaleout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover). For information about changing GTID mode on servers online, see [Section 17.1.4, “Changing GTID Mode on Online Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online).

The key steps in this startup process for the simplest possible GTID replication topology, consisting of one source and one replica, are as follows:

If replication is already running, synchronize both servers by making them read-only.

Stop both servers.

Restart both servers with GTIDs enabled and the correct options configured.

The [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) options necessary to start the servers as described are discussed in the example that follows later in this section.

Instruct the replica to use the source as the replication data source and to use auto-positioning. The SQL statements needed to accomplish this step are described in the example that follows later in this section.

Take a new backup. Binary logs containing transactions without GTIDs cannot be used on servers where GTIDs are enabled, so backups taken before this point cannot be used with your new configuration.

Start the replica, then disable read-only mode on both servers, so that they can accept updates.

In the following example, two servers are already running as source and replica, using MySQL's binary log position-based replication protocol. If you are starting with new servers, see [Section 17.1.2.3, “Creating a User for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-repuser) for information about adding a specific user for replication connections and [Section 17.1.2.1, “Setting the Replication Source Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterbaseconfig) for information about setting the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) variable. The following examples show how to store [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) startup options in server's option file, see [Section 4.2.2.2, “Using Option Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files) for more information. Alternatively you can use startup options when running [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld).

Most of the steps that follow require the use of the MySQL **root** account or another MySQL user account that has the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) **shutdown** requires either the **SUPER** privilege or the [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_shutdown) privilege.

**Step 1: Synchronize the servers.** This step is only required when working with servers which are already replicating without using GTIDs. For new servers proceed to Step 3. Make the servers read-only by setting the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable to **ON** on each server by issuing the following:

mysql> **SET @@GLOBAL.read\_only = ON;**

Wait for all ongoing transactions to commit or roll back. Then, allow the replica to catch up with the source. *It is extremely important that you make sure the replica has processed all updates before continuing*.

If you use binary logs for anything other than replication, for example to do point in time backup and restore, wait until you do not need the old binary logs containing transactions without GTIDs. Ideally, wait for the server to purge all binary logs, and wait for any existing backup to expire.

**Important**

It is important to understand that logs containing transactions without GTIDs cannot be used on servers where GTIDs are enabled. Before proceeding, you must be sure that transactions without GTIDs do not exist anywhere in the topology.

**Step 2: Stop both servers.** Stop each server using [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) as shown here, where ***username*** is the user name for a MySQL user having sufficient privileges to shut down the server:

shell> **mysqladmin -u*username* -p shutdown**

Then supply this user's password at the prompt.

**Step 3: Start both servers with GTIDs enabled.** To enable GTID-based replication, each server must be started with GTID mode enabled by setting the [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) variable to **ON**, and with the [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) variable enabled to ensure that only statements which are safe for GTID-based replication are logged. For example:

gtid\_mode=ON

enforce-gtid-consistency=ON

Start each replica with the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable, to ensure that replication does not start until you have configured the replica settings. For more information on GTID related options and variables, see [Section 17.1.6.5, “Global Transaction ID System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-gtids).

It is not mandatory to have binary logging enabled in order to use GTIDs when using the [mysql.gtid\_executed Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-executed-table). Source servers must always have binary logging enabled in order to be able to replicate. However, replica servers can use GTIDs but without binary logging. If you need to disable binary logging on a replica server, you can do this by specifying the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) and [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) options for the replica.

**Step 4: Configure the replica to use GTID-based auto-positioning.** Tell the replica to use the source with GTID based transactions as the replication data source, and to use GTID-based auto-positioning rather than file-based positioning. Issue a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) on the replica, including the **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option in the statement to tell the replica that the source's transactions are identified by GTIDs.

You may also need to supply appropriate values for the source's host name and port number as well as the user name and password for a replication user account which can be used by the replica to connect to the source; if these have already been set prior to Step 1 and no further changes need to be made, the corresponding options can safely be omitted from the statement shown here.

mysql> **CHANGE MASTER TO**

> **MASTER\_HOST = *host*,**

> **MASTER\_PORT = *port*,**

> **MASTER\_USER = *user*,**

> **MASTER\_PASSWORD = *password*,**

> **MASTER\_AUTO\_POSITION = 1;**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO**

> **SOURCE\_HOST = *host*,**

> **SOURCE\_PORT = *port*,**

> **SOURCE\_USER = *user*,**

> **SOURCE\_PASSWORD = *password*,**

> **SOURCE\_AUTO\_POSITION = 1;**

**Step 5: Take a new backup.** Existing backups that were made before you enabled GTIDs can no longer be used on these servers now that you have enabled GTIDs. Take a new backup at this point, so that you are not left without a usable backup.

For instance, you can execute [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) on the server where you are taking backups. Then either explicitly take a backup or wait for the next iteration of any periodic backup routine you may have set up.

**Step 6: Start the replica and disable read-only mode.** Start the replica like this:

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **START REPLICA;**

The following step is only necessary if you configured a server to be read-only in Step 1. To allow the server to begin accepting updates again, issue the following statement:

mysql> **SET @@GLOBAL.read\_only = OFF;**

GTID-based replication should now be running, and you can begin (or resume) activity on the source as before. [Section 17.1.3.5, “Using GTIDs for Failover and Scaleout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover), discusses creation of new replicas when using GTIDs.

**17.1.3.5 Using GTIDs for Failover and Scaleout**

There are a number of techniques when using MySQL Replication with Global Transaction Identifiers (GTIDs) for provisioning a new replica which can then be used for scaleout, being promoted to source as necessary for failover. This section describes the following techniques:

[Simple replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover-replicate)

[Copying data and transactions to the replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover-copy)

[Injecting empty transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover-empty)

[Excluding transactions with gtid\_purged](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover-gtid-purged)

[Restoring GTID mode replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restoring-mysqlbinlog)

Global transaction identifiers were added to MySQL Replication for the purpose of simplifying in general management of the replication data flow and of failover activities in particular. Each identifier uniquely identifies a set of binary log events that together make up a transaction. GTIDs play a key role in applying changes to the database: the server automatically skips any transaction having an identifier which the server recognizes as one that it has processed before. This behavior is critical for automatic replication positioning and correct failover.

The mapping between identifiers and sets of events comprising a given transaction is captured in the binary log. This poses some challenges when provisioning a new server with data from another existing server. To reproduce the identifier set on the new server, it is necessary to copy the identifiers from the old server to the new one, and to preserve the relationship between the identifiers and the actual events. This is neccessary for restoring a replica that is immediately available as a candidate to become a new source on failover or switchover.

**Simple replication.** The easiest way to reproduce all identifiers and transactions on a new server is to make the new server into the replica of a source that has the entire execution history, and enable global transaction identifiers on both servers. See [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto), for more information.

Once replication is started, the new server copies the entire binary log from the source and thus obtains all information about all GTIDs.

This method is simple and effective, but requires the replica to read the binary log from the source; it can sometimes take a comparatively long time for the new replica to catch up with the source, so this method is not suitable for fast failover or restoring from backup. This section explains how to avoid fetching all of the execution history from the source by copying binary log files to the new server.

**Copying data and transactions to the replica.** Executing the entire transaction history can be time-consuming when the source server has processed a large number of transactions previously, and this can represent a major bottleneck when setting up a new replica. To eliminate this requirement, a snapshot of the data set, the binary logs and the global transaction information the source server contains can be imported to the new replica. The server where the snapshot is taken can be either the source or one of its replicas, but you must ensure that the server has processed all required transactions before copying the data.

There are several variants of this method, the difference being in the manner in which data dumps and transactions from binary logs are transfered to the replica, as outlined here:

**Data Set**

Create a dump file using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) on the source server. Set the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) option [--master-data](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_master-data) (with the default value of 1) to include a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement with binary logging information. Set the [--set-gtid-purged](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_set-gtid-purged) option to **AUTO** (the default) or **ON**, to include information about executed transactions in the dump. Then use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to import the dump file on the target server.

Alternatively, create a data snapshot of the source server using raw data files, then copy these files to the target server, following the instructions in [Section 17.1.2.5, “Choosing a Method for Data Snapshots”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-snapshot-method). If you use [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, you can use the **mysqlbackup** command from the MySQL Enterprise Backup component to produce a consistent snapshot. This command records the log name and offset corresponding to the snapshot to be used on the replica. MySQL Enterprise Backup is a commercial product that is included as part of a MySQL Enterprise subscription. See [Section 30.2, “MySQL Enterprise Backup Overview”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-enterprise.html#mysql-enterprise-backup) for detailed information.

Alternatively, stop both the source and target servers, copy the contents of the source's data directory to the new replica's data directory, then restart the replica. If you use this method, the replica must be configured for GTID-based replication, in other words with [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode). For instructions and important information for this method, see [Section 17.1.2.8, “Adding Replicas to a Replication Environment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-additionalslaves).

**Transaction History**

If the source server has a complete transaction history in its binary logs (that is, the GTID set **@@GLOBAL.gtid\_purged** is empty), you can use these methods.

Import the binary logs from the source server to the new replica using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), with the [--read-from-remote-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_read-from-remote-server) and [--read-from-remote-master](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_read-from-remote-master) options.

Alternatively, copy the source server's binary log files to the replica. You can make copies from the replica using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) with the [--read-from-remote-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_read-from-remote-server) and [--raw](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_raw) options. These can be read into the replica by using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) **>** ***file*** (without the [--raw](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_raw) option) to export the binary log files to SQL files, then passing these files to the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client for processing. Ensure that all of the binary log files are processed using a single [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) process, rather than multiple connections. For example:

shell> **mysqlbinlog copied-binlog.000001 copied-binlog.000002 | mysql -u root -p**

For more information, see [Section 4.6.9.3, “Using mysqlbinlog to Back Up Binary Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog-backup).

This method has the advantage that a new server is available almost immediately; only those transactions that were committed while the snapshot or dump file was being replayed still need to be obtained from the existing source. This means that the replica's availability is not instantanteous, but only a relatively short amount of time should be required for the replica to catch up with these few remaining transactions.

Copying over binary logs to the target server in advance is usually faster than reading the entire transaction execution history from the source in real time. However, it may not always be feasible to move these files to the target when required, due to size or other considerations. The two remaining methods for provisioning a new replica discussed in this section use other means to transfer information about transactions to the new replica.

**Injecting empty transactions.** The source's global [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) variable contains the set of all transactions executed on the source. Rather than copy the binary logs when taking a snapshot to provision a new server, you can instead note the content of **gtid\_executed** on the server from which the snapshot was taken. Before adding the new server to the replication chain, simply commit an empty transaction on the new server for each transaction identifier contained in the source's **gtid\_executed**, like this:

SET GTID\_NEXT='aaa-bbb-ccc-ddd:N';

BEGIN;

COMMIT;

SET GTID\_NEXT='AUTOMATIC';

Once all transaction identifiers have been reinstated in this way using empty transactions, you must flush and purge the replica's binary logs, as shown here, where ***N*** is the nonzero suffix of the current binary log file name:

FLUSH LOGS;

PURGE BINARY LOGS TO 'source-bin.00000***N***';

You should do this to prevent this server from flooding the replication stream with false transactions in the event that it is later promoted to the source. (The [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) statement forces the creation of a new binary log file; [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) purges the empty transactions, but retains their identifiers.)

This method creates a server that is essentially a snapshot, but in time is able to become a source as its binary log history converges with that of the replication stream (that is, as it catches up with the source or sources). This outcome is similar in effect to that obtained using the remaining provisioning method, which we discuss in the next few paragraphs.

**Excluding transactions with gtid\_purged.** The source's global [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) variable contains the set of all transactions that have been purged from the source's binary log. As with the method discussed previously (see [Injecting empty transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover-empty)), you can record the value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) on the server from which the snapshot was taken (in place of copying the binary logs to the new server). Unlike the previous method, there is no need to commit empty transactions (or to issue [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs)); instead, you can set [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) on the replica directly, based on the value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) on the server from which the backup or snapshot was taken.

As with the method using empty transactions, this method creates a server that is functionally a snapshot, but in time is able to become a source as its binary log history converges with that of the source and other replicas.

**Restoring GTID mode replicas.** When restoring a replica in a GTID based replication setup that has encountered an error, injecting an empty transaction may not solve the problem because an event does not have a GTID.

Use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to find the next transaction, which is probably the first transaction in the next log file after the event. Copy everything up to the **COMMIT** for that transaction, being sure to include the **SET @@SESSION.gtid\_next**. Even if you are not using row-based replication, you can still run binary log row events in the command line client.

Stop the replica and run the transaction you copied. The [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output sets the delimiter to **/\*!\*/;**, so set it back:

mysql> **DELIMITER ;**

Restart replication from the correct position automatically:

mysql> **SET GTID\_NEXT=automatic;**

mysql> **RESET SLAVE;**

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **SET GTID\_NEXT=automatic;**

mysql> **RESET REPLICA;**

mysql> **START REPLICA;**

**17.1.3.6 Replication From a Source Without GTIDs to a Replica With GTIDs**

From MySQL 8.0.23, you can set up replication channels to assign a GTID to replicated transactions that do not already have one. This feature enables replication from a source server that does not have GTIDs enabled and does not use GTID-based replication, to a replica that has GTIDs enabled. If it is possible to enable GTIDs on the replication source server, as described in [Section 17.1.4, “Changing GTID Mode on Online Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online), use that approach instead. This feature is designed for replication source servers where you cannot enable GTIDs. Note that as is standard for MySQL replication, this feature does not support replication from MySQL source servers earlier than the previous release series, so MySQL 5.7 is the earliest supported source for a MySQL 8.0 replica.

You can enable GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement. **LOCAL** assigns a GTID including the replica's own UUID (the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting). ***uuid*** assigns a GTID including the specified UUID, such as the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting for the replication source server. Using a nonlocal UUID lets you differentiate between transactions that originated on the replica and transactions that originated on the source, and for a multi-source replica, between transactions that originated on different sources. If any of the transactions sent by the source do have a GTID already, that GTID is retained.

**Important**

A replica set up with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel cannot be promoted to replace the replication source server in the event that a failover is required, and a backup taken from the replica cannot be used to restore the replication source server. The same restriction applies to replacing or restoring other replicas that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel.

The replica must have [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set, and this cannot be changed afterwards, unless you remove the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS=ON** setting. If the replica server is started without GTIDs enabled and with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** set for any replication channels, the settings are not changed, but a warning message is written to the error log explaining how to change the situation.

For a multi-source replica, you can have a mix of channels that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, and channels that do not. Channels specific to Group Replication cannot use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, but an asynchronous replication channel for another source on a server instance that is a Group Replication group member can do so. For a channel on a Group Replication group member, do not specify the Group Replication group name as the UUID for creating the GTIDs.

Using **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on a replication channel is not the same as introducing GTID-based replication for the channel. The GTID set ([**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed)) from a replica set up with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** should not be transferred to another server or compared with another server's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set. The GTIDs that are assigned to the anonymous transactions, and the UUID you choose for them, only have significance for that replica's own use. The exception to this is any downstream replicas of the replica where you enabled **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, and any servers that were created from a backup of that replica.

If you set up any downstream replicas, these servers do not have **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** enabled. Only the replica that is receiving transactions directly from the non-GTID source server needs to have **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** set on the relevant replication channel. Among that replica and its downstream replicas, you can compare GTID sets, fail over from one replica to another, and use backups to create additional replicas, as you would in any GTID-based replication topology. **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** is used where transactions are received from a non-GTID server outside this group.

A replication channel using **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** has the following behavior differences to GTID-based replication:

GTIDs are assigned to the replicated transactions when they are applied (unless they already had a GTID). A GTID would normally be assigned on the replication source server when the transaction is committed, and sent to the replica along with the transaction. On a multi-threaded replica, this means the order of the GTIDs does not necessarily match the order of the transactions, even if [**slave-preserve-commit-order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) is set.

The **SOURCE\_LOG\_FILE** and **SOURCE\_LOG\_POS** options of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement are used to position the replication I/O thread, rather than the **MASTER\_AUTO\_POSITION** option.

The **SET GLOBAL sql\_slave\_skip\_counter** statement is used to skip transactions on a replication channel set up with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, rather than the method of committing empty transactions. For instructions, see [Section 17.1.7.3, “Skipping Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip).

The **UNTIL SQL\_BEFORE\_GTIDS** and **UNTIL\_SQL\_AFTER\_GTIDS** options of the [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement cannot be used for the channel.

The function **WAIT\_UNTIL\_SQL\_THREAD\_AFTER\_GTIDS()**, which is deprecated from MySQL 8.0.18, cannot be used with the channel. Its replacement **WAIT\_FOR\_EXECUTED\_GTID\_SET()**, which works across the server, can be used to wait for any downstream replicas of the server that has **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** enabled. To wait for the channel with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** enabled to catch up with the source, which does not use GTIDs, use the **MASTER\_POS\_WAIT()** function.

The Performance Schema table [**replication\_applier\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-configuration-table) shows whether GTIDs are assigned to anonymous transactions on a replication channel, what the UUID is, and whether it is the UUID of the replica server (**LOCAL**) or a user-specified UUID (**MANUAL**). The information is also recorded in the applier metadata repository. A [**RESET SLAVE ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave) statement resets the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** setting, but a [**RESET SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave) statement does not.

**17.1.3.7 Restrictions on Replication with GTIDs**

Because GTID-based replication is dependent on transactions, some features otherwise available in MySQL are not supported when using it. This section provides information about restrictions on and limitations of replication with GTIDs.

**Updates involving nontransactional storage engines.** When using GTIDs, updates to tables using nontransactional storage engines such as [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) cannot be made in the same statement or transaction as updates to tables using transactional storage engines such as [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html).

This restriction is due to the fact that updates to tables that use a nontransactional storage engine mixed with updates to tables that use a transactional storage engine within the same transaction can result in multiple GTIDs being assigned to the same transaction.

Such problems can also occur when the source and the replica use different storage engines for their respective versions of the same table, where one storage engine is transactional and the other is not. Also be aware that triggers that are defined to operate on nontransactional tables can be the cause of these problems.

In any of the cases just mentioned, the one-to-one correspondence between transactions and GTIDs is broken, with the result that GTID-based replication cannot function correctly.

**CREATE TABLE ... SELECT statements.** Prior to MySQL 8.0.21, [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements are not allowed when using GTID-based replication. When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**, a [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement is recorded in the binary log as one transaction with one GTID, but if **ROW** format is used, the statement is recorded as two transactions with two GTIDs. If a source used **STATEMENT** format and a replica used **ROW** format, the replica would be unable to handle the transaction correctly, therefore the [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement is disallowed with GTIDs to prevent this scenario. This restriction is lifted in MySQL 8.0.21 on storage engines that support atomic DDL. In this case, [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) is recorded in the binary log as one transaction. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

**Temporary tables.** When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**, [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**DROP TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements cannot be used inside transactions, procedures, functions, and triggers when GTIDs are in use on the server (that is, when the [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) system variable is set to **ON**). They can be used outside these contexts when GTIDs are in use, provided that [**autocommit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is set. From MySQL 8.0.13, when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **ROW** or **MIXED**, [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**DROP TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements are allowed inside a transaction, procedure, function, or trigger when GTIDs are in use. The statements are not written to the binary log and are therefore not replicated to replicas. The use of row-based replication means that the replicas remain in sync without the need to replicate temporary tables. If the removal of these statements from a transaction results in an empty transaction, the transaction is not written to the binary log.

**Preventing execution of unsupported statements.** To prevent execution of statements that would cause GTID-based replication to fail, all servers must be started with the [--enforce-gtid-consistency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) option when enabling GTIDs. This causes statements of any of the types discussed previously in this section to fail with an error.

Note that [--enforce-gtid-consistency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) only takes effect if binary logging takes place for a statement. If binary logging is disabled on the server, or if statements are not written to the binary log because they are removed by a filter, GTID consistency is not checked or enforced for the statements that are not logged.

For information about other required startup options when enabling GTIDs, see [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto).

**Skipping transactions.** [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) is not available when using GTID-based replication. If you need to skip transactions, use the value of the source's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) variable instead. If you have enabled GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement, [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) is available. For more information, see [Section 17.1.7.3, “Skipping Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip).

**Ignoring servers.** The IGNORE\_SERVER\_IDS option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement is deprecated when using GTIDs, because transactions that have already been applied are automatically ignored. Before starting GTID-based replication, check for and clear all ignored server ID lists that have previously been set on the servers involved. The [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement, which can be issued for individual channels, displays the list of ignored server IDs if there is one. If there is no list, the **Replicate\_Ignore\_Server\_Ids** field is blank.

**GTID mode and mysql\_upgrade.** Prior to MySQL 8.0.16, when the server is running with global transaction identifiers (GTIDs) enabled ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), do not enable binary logging by [**mysql\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade) (the [--write-binlog](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_upgrade_write-binlog) option). As of MySQL 8.0.16, the server performs the entire MySQL upgrade procedure, but disables binary logging during the upgrade, so there is no issue.

**17.1.3.8 Stored Function Examples to Manipulate GTIDs**

MySQL includes some built-in (native) functions for use with GTID-based replication. These functions are as follows:

GTID\_SUBSET(***set1***,***set2***)

Given two sets of global transaction identifiers ***set1*** and ***set2***, returns true if all GTIDs in ***set1*** are also in ***set2***. Returns false otherwise.

GTID\_SUBTRACT(***set1***,***set2***)

Given two sets of global transaction identifiers ***set1*** and ***set2***, returns only those GTIDs from ***set1*** that are not in ***set2***.

WAIT\_FOR\_EXECUTED\_GTID\_SET(***gtid\_set***[, ***timeout***])

Wait until the server has applied all of the transactions whose global transaction identifiers are contained in ***gtid\_set***. The optional timeout stops the function from waiting after the specified number of seconds have elapsed.

For details of these functions, see [Section 12.19, “Functions Used with Global Transaction Identifiers (GTIDs)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#gtid-functions).

You can define your own stored functions to work with GTIDs. For information on defining stored functions, see [Chapter 25, *Stored Objects*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html). The following examples show some useful stored functions that can be created based on the built-in GTID\_SUBSET() and GTID\_SUBTRACT() functions.

Note that in these stored functions, the delimiter command has been used to change the MySQL statement delimiter to a vertical bar, as follows:

mysql> delimiter |

All of these functions take string representations of GTID sets as arguments, so GTID sets must always be quoted when used with them.

This function returns nonzero (true) if two GTID sets are the same set, even if they are not formatted in the same way.

CREATE FUNCTION GTID\_IS\_EQUAL(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT)

RETURNS INT

RETURN GTID\_SUBSET(gtid\_set\_1, gtid\_set\_2) AND GTID\_SUBSET(gtid\_set\_2, gtid\_set\_1)|

This function returns nonzero (true) if two GTID sets are disjoint.

CREATE FUNCTION GTID\_IS\_DISJOINT(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT)

RETURNS INT

RETURN GTID\_SUBSET(gtid\_set\_1, GTID\_SUBTRACT(gtid\_set\_1, gtid\_set\_2))|

This function returns nonzero (true) if two GTID sets are disjoint, and **sum** is the union of the two sets.

CREATE FUNCTION GTID\_IS\_DISJOINT\_UNION(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT, sum LONGTEXT)

RETURNS INT

RETURN GTID\_IS\_EQUAL(GTID\_SUBTRACT(sum, gtid\_set\_1), gtid\_set\_2) AND

GTID\_IS\_EQUAL(GTID\_SUBTRACT(sum, gtid\_set\_2), gtid\_set\_1)|

This function returns a normalized form of the GTID set, in all uppercase, with no whitespace and no duplicates. The UUIDs are arranged in alphabetic order and intervals are arranged in numeric order.

CREATE FUNCTION GTID\_NORMALIZE(g LONGTEXT)

RETURNS LONGTEXT

RETURN GTID\_SUBTRACT(g, '')|

This function returns the union of two GTID sets.

CREATE FUNCTION GTID\_UNION(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT)

RETURNS LONGTEXT

RETURN GTID\_NORMALIZE(CONCAT(gtid\_set\_1, ',', gtid\_set\_2))|

This function returns the intersection of two GTID sets.

CREATE FUNCTION GTID\_INTERSECTION(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT)

RETURNS LONGTEXT

RETURN GTID\_SUBTRACT(gtid\_set\_1, GTID\_SUBTRACT(gtid\_set\_1, gtid\_set\_2))|

This function returns the symmetric difference between two GTID sets, that is, the GTIDs that exist in **gtid\_set\_1** but not in **gtid\_set\_2**, and also the GTIDs that exist in **gtid\_set\_2** but not in **gtid\_set\_1**.

CREATE FUNCTION GTID\_SYMMETRIC\_DIFFERENCE(gtid\_set\_1 LONGTEXT, gtid\_set\_2 LONGTEXT)

RETURNS LONGTEXT

RETURN GTID\_SUBTRACT(CONCAT(gtid\_set\_1, ',', gtid\_set\_2), GTID\_INTERSECTION(gtid\_set\_1, gtid\_set\_2))|

This function removes from a GTID set all the GTIDs from a specified origin, and returns the remaining GTIDs, if any. The UUID is the identifier used by the server where the transaction originated, which is normally the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) value.

CREATE FUNCTION GTID\_SUBTRACT\_UUID(gtid\_set LONGTEXT, uuid TEXT)

RETURNS LONGTEXT

RETURN GTID\_SUBTRACT(gtid\_set, CONCAT(UUID, ':1-', (1 << 63) - 2))|

This function reverses the previously listed function to return only those GTIDs from the GTID set that originate from the server with the specified identifier (UUID).

CREATE FUNCTION GTID\_INTERSECTION\_WITH\_UUID(gtid\_set LONGTEXT, uuid TEXT)

RETURNS LONGTEXT

RETURN GTID\_SUBTRACT(gtid\_set, GTID\_SUBTRACT\_UUID(gtid\_set, uuid))|

**Example 17.1 Verifying that a replica is up to date**

The built-in functions GTID\_SUBSET and GTID\_SUBTRACT can be used to check that a replica has applied at least every transaction that a source has applied.

To perform this check with GTID\_SUBSET, execute the following statement on the replica:

SELECT GTID\_SUBSET(***source\_gtid\_executed***, ***replica\_gtid\_executed***)

If this returns 0 (false), some GTIDs in ***source\_gtid\_executed*** are not present in ***replica\_gtid\_executed***, so the source has applied some transactions that the replica has not applied, and the replica is therefore not up to date.

To perform the check with GTID\_SUBTRACT, execute the following statement on the replica:

SELECT GTID\_SUBTRACT(***source\_gtid\_executed***, ***replica\_gtid\_executed***)

This statement returns any GTIDs that are in ***source\_gtid\_executed*** but not in ***replica\_gtid\_executed***. If any GTIDs are returned, the source has applied some transactions that the replica has not applied, and the replica is therefore not up to date.

**Example 17.2 Backup and restore scenario**

The stored functions GTID\_IS\_EQUAL, GTID\_IS\_DISJOINT, and GTID\_IS\_DISJOINT\_UNION could be used to verify backup and restore operations involving multiple databases and servers. In this example scenario, **server1** contains database **db1**, and **server2** contains database **db2**. The goal is to copy database **db2** to **server1**, and the result on **server1** should be the union of the two databases. The procedure used is to back up **server2** using [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump) or [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump), then restore this backup on **server1**.

Provided the backup program's option --set-gtid-purged was set to **ON** or the default of **AUTO**, the program's output contains a **SET @@GLOBAL.gtid\_purged** statement which adds the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set from **server2** to the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set on **server1**. The [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set contains the GTIDs of all the transactions that have been committed on a server but do not exist in any binary log file on the server. When database **db2** is copied to **server1**, the GTIDs of the transactions committed on **server2**, which are not in the binary log files on **server1**, must be added to **server1**'s [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set to make the set complete.

The stored functions can be used to assist with the following steps in this scenario:

Use GTID\_IS\_EQUAL to verify that the backup operation computed the correct GTID set for the **SET @@GLOBAL.gtid\_purged** statement. On **server2**, extract that statement from the [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump) or [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) output, and store the GTID set into a local variable, such as $gtid\_purged\_set. Then execute the following statement:

server2> SELECT GTID\_IS\_EQUAL($gtid\_purged\_set, @@GLOBAL.gtid\_executed);

If the result is 1, the two GTID sets are equal, and the set has been computed correctly.

Use GTID\_IS\_DISJOINT to verify that the GTID set in the [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump) or [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) output does not overlap with the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on **server1**. Having identical GTIDs present on both servers causes errors when copying database **db2** to **server1**. To check, on **server1**, extract and store the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set from the output into a local variable as above, then execute the following statement:

server1> SELECT GTID\_IS\_DISJOINT($gtid\_purged\_set, @@GLOBAL.gtid\_executed);

If the result is 1, there is no overlap between the two GTID sets, so no duplicate GTIDs are present.

Use GTID\_IS\_DISJOINT\_UNION to verify that the restore operation resulted in the correct GTID state on **server1**. Before restoring the backup, on **server1**, obtain the existing [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set by executing the following statement:

server1> SELECT @@GLOBAL.gtid\_executed;

Store the result in a local variable $original\_gtid\_executed. Also store the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set in a local variable as described above. When the backup from **server2** has been restored onto **server1**, execute the following statement to verify the GTID state:

server1> SELECT GTID\_IS\_DISJOINT\_UNION($original\_gtid\_executed,

                                       $gtid\_purged\_set,

                                       @@GLOBAL.gtid\_executed);

If the result is 1, the stored function has verified that the original [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set from **server1** ($original\_gtid\_executed) and the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set that was added from **server2** ($gtid\_purged\_set) have no overlap, and also that the updated [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on **server1** now consists of the previous [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set from **server1** plus the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set from **server2**, which is the desired result. Ensure that this check is carried out before any further transactions take place on **server1**, otherwise the new transactions in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set cause it to fail.

**Example 17.3 Selecting the most up-to-date replica for manual failover**

The stored function GTID\_UNION could be used to identify the most up-to-date replica from a set of replicas, in order to perform a manual failover operation after a source server has stopped unexpectedly. If some of the replicas are experiencing replication lag, this stored function can be used to compute the most up-to-date replica without waiting for all the replicas to apply their existing relay logs, and therefore to minimize the failover time. The function can return the union of the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on each replica with the set of transactions received by the replica, which is recorded in the Performance Schema table [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table). You can compare these results to find which replica's record of transactions is the most up-to-date, even if not all of the transactions have been committed yet.

On each replica, compute the complete record of transactions by issuing the following statement:

SELECT GTID\_UNION(RECEIVED\_TRANSACTION\_SET, @@GLOBAL.gtid\_executed)

    FROM performance\_schema.replication\_connection\_status

    WHERE channel\_name = 'name';

You can then compare the results from each replica to see which one has the most up-to-date record of transactions, and use this replica as the new source.

**Example 17.4 Checking for extraneous transactions on a replica**

The stored function GTID\_SUBTRACT\_UUID could be used to check whether a replica has received transactions that did not originate from its designated source or sources. If it has, there might be an issue with your replication setup, or with a proxy, router, or load balancer. This function works by removing from a GTID set all the GTIDs from a specified originating server, and returning the remaining GTIDs, if any.

For a replica that replicates from a single source, issue the following statement, giving the identifier of the originating source, which is normally the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) value:

SELECT GTID\_SUBTRACT\_UUID(@@GLOBAL.gtid\_executed, server\_uuid\_of\_source);

  If the result is not empty, the transactions returned are extra transactions that did not originate from the designated source.

For a replica in a multisource replication topology, repeat the function, for example:

SELECT GTID\_SUBTRACT\_UUID(GTID\_SUBTRACT\_UUID(@@GLOBAL.gtid\_executed,

                      server\_uuid\_of\_source\_1),

                     server\_uuid\_of\_source\_2);

If the result is not empty, the transactions returned are extra transactions that did not originate from any of the designated sources.

**Example 17.5 Verifying that a server in a replication topology is read-only**

The stored function GTID\_INTERSECTION\_WITH\_UUID could be used to verify that a server has not originated any GTIDs and is in a read-only state. The function returns only those GTIDs from the GTID set that originate from the server with the specified identifier. If any of the transactions in the server's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set have the server's own identifier, the server itself originated those transactions. You can issue the following statement on the server to check:

SELECT GTID\_INTERSECTION\_WITH\_UUID(@@GLOBAL.gtid\_executed, my\_server\_uuid);

**Example 17.6 Validating an additional replica in a multisource replication setup**

The stored function GTID\_INTERSECTION\_WITH\_UUID could be used to find out if a replica attached to a multisource replication setup has applied all the transactions originating from one particular source. In this scenario, **source1** and **source2** are both sources and replicas and replicate to each other. **source2** also has its own replica. The replica also receives and applies transactions from source **source1** if **source2** is configured with [**log\_slave\_updates=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates), but it does not do so if **source2** uses [**log\_slave\_updates=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates). Whatever the case, we currently only want to find out if the replica is up to date with **source2**. In this situation, the stored function GTID\_INTERSECTION\_WITH\_UUID can be used to identify the transactions that **source2** originated, discarding the transactions that **source2** has replicated from **source1**. The built-in function GTID\_SUBSET can then be used to compare the result to the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on the replica. If the replica is up to date with **source2**, the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on the replica contains all the transactions in the intersection set (the transactions that originated from **source2**).

To carry out this check, store **source2**'s [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set, **source2**'s server UUID, and the replica's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set, into client-side variables as follows:

$source2\_gtid\_executed :=

source2> SELECT @@GLOBAL.gtid\_executed;

$source2\_server\_uuid :=

source2> SELECT @@GLOBAL.server\_uuid;

$replica\_gtid\_executed :=

replica> SELECT @@GLOBAL.gtid\_executed;

Then use GTID\_INTERSECTION\_WITH\_UUID and GTID\_SUBSET with these variables as input, as follows:

SELECT GTID\_SUBSET(GTID\_INTERSECTION\_WITH\_UUID($source2\_gtid\_executed,

$source2\_server\_uuid),

$replica\_gtid\_executed);

The server identifier from **source2** ($source2\_server\_uuid) is used with **GTID\_INTERSECTION\_WITH\_UUID** to identify and return only those GTIDs from **source2**'s [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set that originated on **source2**, omitting those that originated on **source1**. The resulting GTID set is then compared with the set of all executed GTIDs on the replica, using GTID\_SUBSET. If this statement returns nonzero (true), all the identified GTIDs from **source2** (the first set input) are also in the replica's [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set (the second set input), meaning that the replica has replicated all the transactions that originated from **source2**.

**17.1.4 Changing GTID Mode on Online Servers**

[17.1.4.1 Replication Mode Concepts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-concepts)

[17.1.4.2 Enabling GTID Transactions Online](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-enable-gtids)

[17.1.4.3 Disabling GTID Transactions Online](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-disable-gtids)

[17.1.4.4 Verifying Replication of Anonymous Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-verify-transactions)

This section describes how to change the mode of replication from and to GTID mode without having to take the server offline.

**17.1.4.1 Replication Mode Concepts**

To be able to safely configure the replication mode of an online server it is important to understand some key concepts of replication. This section explains these concepts and is essential reading before attempting to modify the replication mode of an online server.

The modes of replication available in MySQL rely on different techniques for identifying transactions which are logged. The types of transactions used by replication are as follows:

GTID transactions are identified by a global transaction identifier (GTID) in the form **UUID:NUMBER**. Every GTID transaction in a log is always preceded by a **Gtid\_log\_event**. GTID transactions can be addressed using either the GTID or using the file name and position.

Anonymous transactions do not have a GTID assigned, and MySQL ensures that every anonymous transaction in a log is preceded by an **Anonymous\_gtid\_log\_event**. In previous versions, anonymous transactions were not preceded by any particular event. Anonymous transactions can only be addressed using file name and position.

When using GTIDs you can take advantage of GTID auto-positioning and automatic fail-over, as well as use [**WAIT\_FOR\_EXECUTED\_GTID\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_wait-for-executed-gtid-set), [**session\_track\_gtids**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_session_track_gtids), and monitor replicated transactions using Performance Schema tables.

Transactions in a relay log that was received from a source running a previous version of MySQL may not be preceded by any particular event at all, but after being replayed and logged in the replica's binary log, they are preceded with an **Anonymous\_gtid\_log\_event**.

The ability to configure the replication mode online means that the [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) and [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) variables are now both dynamic and can be set from a top-level statement by an account that has privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges). In MySQL 5.6 and earlier, both of these variables could only be configured using the appropriate option at server start, meaning that changes to the replication mode required a server restart. In all versions [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) could be set to **ON** or **OFF**, which corresponded to whether GTIDs were used to identify transactions or not. When [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) it is not possible to replicate anonymous transactions, and when [**gtid\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) only anonymous transactions can be replicated. When [**gtid\_mode=OFF\_PERMISSIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) then *new* transactions are anonymous while permitting replicated transactions to be either GTID or anonymous transactions. When [**gtid\_mode=ON\_PERMISSIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) then *new* transactions use GTIDs while permitting replicated transactions to be either GTID or anonymous transactions. This means it is possible to have a replication topology that has servers using both anonymous and GTID transactions. For example a source with [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) could be replicating to a replica with [**gtid\_mode=ON\_PERMISSIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode). The valid values for [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) are as follows and in this order:

**OFF**

**OFF\_PERMISSIVE**

**ON\_PERMISSIVE**

**ON**

It is important to note that the state of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) can only be changed by one step at a time based on the above order. For example, if [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is currently set to **OFF\_PERMISSIVE**, it is possible to change to **OFF** or **ON\_PERMISSIVE** but not to **ON**. This is to ensure that the process of changing from anonymous transactions to GTID transactions online is correctly handled by the server. When you switch between [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) and [**gtid\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), the GTID state (in other words the value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed)) is persistent. This ensures that the GTID set that has been applied by the server is always retained, regardless of changes between types of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode).

The fields related to GTIDs display the correct information regardless of the currently selected [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode). This means that fields which display GTID sets, such as [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed), [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), **RECEIVED\_TRANSACTION\_SET** in the [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) Performance Schema table, and the GTID related results of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status), now return the empty string when there are no GTIDs present. Fields that display a single GTID, such as **CURRENT\_TRANSACTION** in the Performance Schema [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) table, now display **ANONYMOUS** when GTID transactions are not being used.

Replication from a source using [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) provides the ability to use GTID auto-positioning, configured using the **SOURCE\_AUTO\_POSITION** of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23), or the **MASTER\_AUTO\_POSITION** option of the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). The replication topology being used impacts on whether it is possible to enable auto-positioning or not, as this feature relies on GTIDs and is not compatible with anonymous transactions. It is strongly recommended to ensure there are no anonymous transactions remaining in the topology before enabling auto-positioning, see [Section 17.1.4.2, “Enabling GTID Transactions Online”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-enable-gtids).

The valid combinations of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) and auto-positioning on source and replica are shown in the following table, where the source's [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is shown on the horizontal and the replica's [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is on the vertical. The meaning of each entry is as follows:

**Y**: the [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) of source and replica is compatible

**N**: the [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) of source and replica is not compatible

**\***: auto-positioning can be used with this combination

**Table 17.1 Valid Combinations of Source and Replica gtid\_mode**

| [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) | **Source OFF** | **Source OFF\_PERMISSIVE** | **Source ON\_PERMISSIVE** | **Source ON** |
| --- | --- | --- | --- | --- |
| **Replica OFF** | Y | Y | N | N |
| **Replica OFF\_PERMISSIVE** | Y | Y | Y | Y\* |
| **Replica ON\_PERMISSIVE** | Y | Y | Y | Y\* |
| **Replica ON** | N | N | Y | Y\* |

The currently selected [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) also impacts on the [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) variable. The following table shows the behavior of the server for the different values of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) and [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next). The meaning of each entry is as follows:

**ANONYMOUS**: generate an anonymous transaction.

**Error**: generate an error and fail to execute **SET GTID\_NEXT**.

**UUID:NUMBER**: generate a GTID with the specified UUID:NUMBER.

**New GTID**: generate a GTID with an automatically generated number.

**Table 17.2 Valid Combinations of gtid\_mode and gtid\_next**

|  | [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next)**AUTOMATIC**  **binary log on** | [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next)**AUTOMATIC**  **binary log off** | [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next)**ANONYMOUS** | [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next)**UUID:NUMBER** |
| --- | --- | --- | --- | --- |
| [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)**OFF** | ANONYMOUS | ANONYMOUS | ANONYMOUS | Error |
| [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)**OFF\_PERMISSIVE** | ANONYMOUS | ANONYMOUS | ANONYMOUS | UUID:NUMBER |
| [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)**ON\_PERMISSIVE** | New GTID | ANONYMOUS | ANONYMOUS | UUID:NUMBER |
| [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)**ON** | New GTID | ANONYMOUS | Error | UUID:NUMBER |

When the binary log is off and [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) is set to **AUTOMATIC**, then no GTID is generated. This is consistent with the behavior of previous versions.

**17.1.4.2 Enabling GTID Transactions Online**

This section describes how to enable GTID transactions, and optionally auto-positioning, on servers that are already online and using anonymous transactions. This procedure does not require taking the server offline and is suited to use in production. However, if you have the possibility to take the servers offline when enabling GTID transactions that process is easier.

From MySQL 8.0.23, you can set up replication channels to assign a GTID to replicated transactions that do not already have one. This feature enables replication from a source server that does not use GTID-based replication, to a replica that does. If it is possible to enable GTIDs on the replication source server, as described in this procedure, use this approach instead. Assigning GTIDs is designed for replication source servers where you cannot enable GTIDs. For more information on this option, see [Section 17.1.3.6, “Replication From a Source Without GTIDs to a Replica With GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-assign-anon).

Before you start, ensure that the servers meet the following pre-conditions:

*All* servers in your topology must use MySQL 5.7.6 or later. You cannot enable GTID transactions online on any single server unless *all* servers which are in the topology are using this version.

All servers have [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set to the default value **OFF**.

The following procedure can be paused at any time and later resumed where it was, or reversed by jumping to the corresponding step of [Section 17.1.4.3, “Disabling GTID Transactions Online”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-disable-gtids), the online procedure to disable GTIDs. This makes the procedure fault-tolerant because any unrelated issues that may appear in the middle of the procedure can be handled as usual, and then the procedure continued where it was left off.

**Note**

It is crucial that you complete every step before continuing to the next step.

To enable GTID transactions:

On each server, execute:

SET @@GLOBAL.ENFORCE\_GTID\_CONSISTENCY = WARN;

Let the server run for a while with your normal workload and monitor the logs. If this step causes any warnings in the log, adjust your application so that it only uses GTID-compatible features and does not generate any warnings.

**Important**

This is the first important step. You must ensure that no warnings are being generated in the error logs before going to the next step.

On each server, execute:

SET @@GLOBAL.ENFORCE\_GTID\_CONSISTENCY = ON;

On each server, execute:

SET @@GLOBAL.GTID\_MODE = OFF\_PERMISSIVE;

It does not matter which server executes this statement first, but it is important that all servers complete this step before any server begins the next step.

On each server, execute:

SET @@GLOBAL.GTID\_MODE = ON\_PERMISSIVE;

It does not matter which server executes this statement first.

On each server, wait until the status variable **ONGOING\_ANONYMOUS\_TRANSACTION\_COUNT** is zero. This can be checked using:

SHOW STATUS LIKE 'ONGOING\_ANONYMOUS\_TRANSACTION\_COUNT';

**Note**

On a replica, it is theoretically possible that this shows zero and then nonzero again. This is not a problem, it suffices that it shows zero once.

Wait for all transactions generated up to step 5 to replicate to all servers. You can do this without stopping updates: the only important thing is that all anonymous transactions get replicated.

See [Section 17.1.4.4, “Verifying Replication of Anonymous Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-verify-transactions) for one method of checking that all anonymous transactions have replicated to all servers.

If you use binary logs for anything other than replication, for example point in time backup and restore, wait until you do not need the old binary logs having transactions without GTIDs.

For instance, after step 6 has completed, you can execute [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) on the server where you are taking backups. Then either explicitly take a backup or wait for the next iteration of any periodic backup routine you may have set up.

Ideally, wait for the server to purge all binary logs that existed when step 6 was completed. Also wait for any backup taken before step 6 to expire.

**Important**

This is the second important point. It is vital to understand that binary logs containing anonymous transactions, without GTIDs cannot be used after the next step. After this step, you must be sure that transactions without GTIDs do not exist anywhere in the topology.

On each server, execute:

SET @@GLOBAL.GTID\_MODE = ON;

On each server, add **gtid\_mode=ON** and **enforce\_gtid\_consistency=ON** to my.cnf.

You are now guaranteed that all transactions have a GTID (except transactions generated in step 5 or earlier, which have already been processed). To start using the GTID protocol so that you can later perform automatic fail-over, execute the following on each replica. Optionally, if you use multi-source replication, do this for each channel and include the **FOR CHANNEL *channel*** clause:

STOP SLAVE [FOR CHANNEL 'channel'];

CHANGE MASTER TO MASTER\_AUTO\_POSITION = 1 [FOR CHANNEL 'channel'];

START SLAVE [FOR CHANNEL 'channel'];

Or from MySQL 8.0.22 / 8.0.23:

STOP REPLICA [FOR CHANNEL 'channel'];

CHANGE REPLICATION SOURCE TO SOURCE\_AUTO\_POSITION = 1 [FOR CHANNEL 'channel'];

START REPLICA [FOR CHANNEL 'channel'];

**17.1.4.3 Disabling GTID Transactions Online**

This section describes how to disable GTID transactions on servers that are already online. This procedure does not require taking the server offline and is suited to use in production. However, if you have the possibility to take the servers offline when disabling GTIDs mode that process is easier.

The process is similar to enabling GTID transactions while the server is online, but reversing the steps. The only thing that differs is the point at which you wait for logged transactions to replicate.

Before you start, ensure that the servers meet the following pre-conditions:

*All* servers in your topology must use MySQL 5.7.6 or later. You cannot disable GTID transactions online on any single server unless *all* servers which are in the topology are using this version.

All servers have [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set to **ON**.

The [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id) option is not set on any server. You cannot disable GTID transactions if this option is set together with the [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) option (which is the default) and binary logging is enabled (which is also the default). Without GTIDs, this combination of options causes infinite loops in circular replication.

Execute the following on each replica, and if you are using multi-source replication, do it for each channel and include the **FOR CHANNEL** channel clause:

STOP SLAVE [FOR CHANNEL 'channel'];

CHANGE MASTER TO MASTER\_AUTO\_POSITION = 0, MASTER\_LOG\_FILE = file, \

MASTER\_LOG\_POS = position [FOR CHANNEL 'channel'];

START SLAVE [FOR CHANNEL 'channel'];

Or from MySQL 8.0.22 / 8.0.23:

STOP REPLICA [FOR CHANNEL 'channel'];

CHANGE REPLICATION SOURCE TO SOURCE\_AUTO\_POSITION = 0, SOURCE\_LOG\_FILE = file, \

SOURCE\_LOG\_POS = position [FOR CHANNEL 'channel'];

START REPLICA [FOR CHANNEL 'channel'];

On each server, execute:

SET @@GLOBAL.GTID\_MODE = ON\_PERMISSIVE;

On each server, execute:

SET @@GLOBAL.GTID\_MODE = OFF\_PERMISSIVE;

On each server, wait until the variable @@GLOBAL.GTID\_OWNED is equal to the empty string. This can be checked using:

SELECT @@GLOBAL.GTID\_OWNED;

On a replica, it is theoretically possible that this is empty and then nonempty again. This is not a problem, it suffices that it is empty once.

Wait for all transactions that currently exist in any binary log to replicate to all replicas. See [Section 17.1.4.4, “Verifying Replication of Anonymous Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-verify-transactions) for one method of checking that all anonymous transactions have replicated to all servers.

If you use binary logs for anything else than replication, for example to do point in time backup or restore: wait until you do not need the old binary logs having GTID transactions.

For instance, after step 5 has completed, you can execute [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) on the server where you are taking the backup. Then either explicitly take a backup or wait for the next iteration of any periodic backup routine you may have set up.

Ideally, wait for the server to purge all binary logs that existed when step 5 was completed. Also wait for any backup taken before step 5 to expire.

**Important**

This is the one important point during this procedure. It is important to understand that logs containing GTID transactions cannot be used after the next step. Before proceeding you must be sure that GTID transactions do not exist anywhere in the topology.

On each server, execute:

SET @@GLOBAL.GTID\_MODE = OFF;

On each server, set [**gtid\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) in my.cnf.

If you want to set [**enforce\_gtid\_consistency=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency), you can do so now. After setting it, you should add [**enforce\_gtid\_consistency=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) to your configuration file.

If you want to downgrade to an earlier version of MySQL, you can do so now, using the normal downgrade procedure.

**17.1.4.4 Verifying Replication of Anonymous Transactions**

This section explains how to monitor a replication topology and verify that all anonymous transactions have been replicated. This is helpful when changing the replication mode online as you can verify that it is safe to change to GTID transactions.

There are several possible ways to wait for transactions to replicate:

The simplest method, which works regardless of your topology but relies on timing is as follows: if you are sure that the replica never lags more than N seconds, just wait for a bit more than N seconds. Or wait for a day, or whatever time period you consider safe for your deployment.

A safer method in the sense that it does not depend on timing: if you only have a source with one or more replicas, do the following:

On the source, execute:

SHOW MASTER STATUS;

Note down the values in the **File** and **Position** column.

On every replica, use the file and position information from the source to execute:

SELECT MASTER\_POS\_WAIT(file, position);

If you have a source and multiple levels of replicas, or in other words you have replicas of replicas, repeat step 2 on each level, starting from the source, then all the direct replicas, then all the replicas of replicas, and so on.

If you use a circular replication topology where multiple servers may have write clients, perform step 2 for each source-replica connection, until you have completed the full circle. Repeat the whole process so that you do the full circle *twice*.

For example, suppose you have three servers A, B, and C, replicating in a circle so that A -> B -> C -> A. The procedure is then:

Do step 1 on A and step 2 on B.

Do step 1 on B and step 2 on C.

Do step 1 on C and step 2 on A.

Do step 1 on A and step 2 on B.

Do step 1 on B and step 2 on C.

Do step 1 on C and step 2 on A.

**17.1.5 MySQL Multi-Source Replication**

[17.1.5.1 Configuring Multi-Source Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-configuration)

[17.1.5.2 Provisioning a Multi-Source Replica for GTID-Based Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-provision-replica)

[17.1.5.3 Adding GTID-Based Sources to a Multi-Source Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-adding-gtid-master)

[17.1.5.4 Adding Binary Log Based Replication Sources to a Multi-Source Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-adding-binlog-master)

[17.1.5.5 Starting Multi-Source Replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-start-replica)

[17.1.5.6 Stopping Multi-Source Replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-stop-replica)

[17.1.5.7 Resetting Multi-Source Replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-reset-replica)

[17.1.5.8 Monitoring Multi-Source Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source-monitoring)

MySQL multi-source replication enables a replica to receive transactions from multiple immediate sources in parallel. In a multi-source replication topology, a replica creates a replication channel for each source that it should receive transactions from. For more information on how replication channels function, see [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels).

You might choose to implement multi-source replication to achieve goals like these:

Backing up multiple servers to a single server.

Merging table shards.

Consolidating data from multiple servers to a single server.

Multi-source replication does not implement any conflict detection or resolution when applying transactions, and those tasks are left to the application if required.

**Note**

Each channel on a multi-source replica must replicate from a different source. You cannot set up multiple replication channels from a single replica to a single source. This is because the server IDs of replicas must be unique in a replication topology. The source distinguishes replicas only by their server IDs, not by the names of the replication channels, so it cannot recognize different replication channels from the same replica.

A rmulti-source replica can also be set up as a multi-threaded replica, by setting the [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) system variable to a value greater than 0. When you do this on a multi-source replica, each channel on the replica has the specified number of applier threads, plus a coordinator thread to manage them. You cannot configure the number of applier threads for individual channels.

From MySQL 8.0, multi-source replicas can be configured with replication filters on specific replication channels. Channel specific replication filters can be used when the same database or table is present on multiple sources, and you only need the replica to replicate it from one source. For GTID-based replication, if the same transaction might arrive from multiple sources (such as in a diamond topology), you must ensure the filtering setup is the same on all channels. For more information, see [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters).

This section provides tutorials on how to configure sources and replicas for multi-source replication, how to start, stop and reset multi-source replicas, and how to monitor multi-source replication.

**17.1.5.1 Configuring Multi-Source Replication**

A multi-source replication topology requires at least two sources and one replica configured. In these tutorials, we assume that you have two sources **source1** and **source2**, and a replica **replicahost**. The replica replicates one database from each of the sources, **db1** from **source1** and **db2** from **source2**.

Sources in a multi-source replication topology can be configured to use either GTID-based replication, or binary log position-based replication. See [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto) for how to configure a source using GTID-based replication. See [Section 17.1.2.1, “Setting the Replication Source Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-masterbaseconfig) for how to configure a source using file position based replication.

Replicas in a multi-source replication topology require **TABLE** repositories for the replica's connection metadata repository and applier metadata repository, which are the default in MySQL 8.0. Multi-source replication is not compatible with the deprecated alternative file repositories.

Create a suitable user account on all the sources that the replica can use to connect. You can use the same account on all the sources, or a different account on each. If you create an account solely for the purposes of replication, that account needs only the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege. For example, to set up a new user, **ted**, that can connect from the replica **replicahost**, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue these statements on each of the sources:

mysql> **CREATE USER 'ted'@'replicahost' IDENTIFIED BY '*password*';**

mysql> **GRANT REPLICATION SLAVE ON \*.\* TO 'ted'@'replicahost';**

For more details, and important information on the default authentication plugin for new users from MySQL 8.0, see [Section 17.1.2.3, “Creating a User for Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-repuser).

**17.1.5.2 Provisioning a Multi-Source Replica for GTID-Based Replication**

If the sources in the multi-source replication topology have existing data, it can save time to provision the replica with the relevant data before starting replication. In a multi-source replication topology, cloning or copying of the data directory cannot be used to provision the replica with data from all of the sources, and you might also want to replicate only specific databases from each source. The best strategy for provisioning such a replica is therefore to use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to create an appropriate dump file on each source, then use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to import the dump file on the replica.

If you are using GTID-based replication, you need to pay attention to the **SET @@GLOBAL.gtid\_purged** statement that [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) places in the dump output. This statement transfers the GTIDs for the transactions executed on the source to the replica, and the replica requires this information. However, for any case more complex than provisioning one new, empty replica from one source, you need to check what effect the statement has in the version of MySQL used by the replica, and handle the statement accordingly. The following guidance summarizes suitable actions, but for more details, see the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) documentation.

The behavior of the **SET @@GLOBAL.gtid\_purged** statement written by [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) is different in releases from MySQL 8.0 compared to MySQL 5.6 and 5.7. In MySQL 5.6 and 5.7, the statement replaces the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) on the replica, and also in those releases that value can only be changed when the replica's record of transactions with GTIDs (the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set) is empty. In a multi-source replication topology, you must therefore remove the **SET @@GLOBAL.gtid\_purged** statement from the dump output before replaying the dump files, because you cannot apply a second or subsequent dump file including this statement. Also note that for MySQL 5.6 and 5.7, this limitation means all the dump files from the sources must be applied in a single operation on a replica with an empty [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set. You can clear a replica's GTID execution history by issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) on the replica, but if you have other, wanted transactions with GTIDs on the replica, choose an alternative method of provisioning from those described in [Section 17.1.3.5, “Using GTIDs for Failover and Scaleout”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-failover).

From MySQL 8.0, the **SET @@GLOBAL.gtid\_purged** statement adds the GTID set from the dump file to the existing [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) set on the replica. The statement can therefore potentially be left in the dump output when you replay the dump files on the replica, and the dump files can be replayed at different times. However, it is important to note that the value that is included by [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) for the **SET @@GLOBAL.gtid\_purged** statement includes the GTIDs of all transactions in the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set on the source, even those that changed suppressed parts of the database, or other databases on the server that were not included in a partial dump. If you replay a second or subsequent dump file on the replica that contains any of the same GTIDs (for example, another partial dump from the same source, or a dump from another source that has overlapping transactions), any **SET @@GLOBAL.gtid\_purged** statement in the second dump file fails, and must therefore be removed from the dump output.

For sources from MySQL 8.0.17, as an alternative to removing the **SET @@GLOBAL.gtid\_purged** statement, you may set [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump)'s **--set-gtid-purged** option to **COMMENTED** to include the statement but commented out, so that it is not actioned when you load the dump file. If you are provisioning the replica with two partial dumps from the same source, and the GTID set in the second dump is the same as the first (so no new transactions have been executed on the source in between the dumps), you can set [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump)'s **--set-gtid-purged** option to **OFF** when you output the second dump file, to omit the statement.

In the following provisioning example, we assume that the **SET @@GLOBAL.gtid\_purged** statement cannot be left in the dump output, and must be removed from the files and handled manually. We also assume that there are no wanted transactions with GTIDs on the replica before provisioning starts.

To create dump files for a database named **db1** on **source1** and a database named **db2** on **source2**, run [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) for **source1** as follows:

**mysqldump -u<*user*> -p<*password*> --single-transaction --triggers --routines --set-gtid-purged=ON --databases db1 > dumpM1.sql**

Then run [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) for **source2** as follows:

**mysqldump -u<*user*> -p<*password*> --single-transaction --triggers --routines --set-gtid-purged=ON --databases db2 > dumpM2.sql**

Record the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) value that [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) added to each of the dump files. For example, for dump files created on MySQL 5.6 or 5.7, you can extract the value like this:

**cat dumpM1.sql | grep GTID\_PURGED | cut -f2 -d'=' | cut -f2 -d$'\''**

**cat dumpM2.sql | grep GTID\_PURGED | cut -f2 -d'=' | cut -f2 -d$'\''**

From MySQL 8.0, where the format has changed, you can extract the value like this:

**cat dumpM1.sql | grep GTID\_PURGED | perl -p0 -e 's#/\\*.\*?\\*/##sg' | cut -f2 -d'=' | cut -f2 -d$'\''**

**cat dumpM2.sql | grep GTID\_PURGED | perl -p0 -e 's#/\\*.\*?\\*/##sg' | cut -f2 -d'=' | cut -f2 -d$'\''**

The result in each case should be a GTID set, for example:

source1: 2174B383-5441-11E8-B90A-C80AA9429562:1-1029

source2: 224DA167-0C0C-11E8-8442-00059A3C7B00:1-2695

Remove the line from each dump file that contains the **SET @@GLOBAL.gtid\_purged** statement. For example:

**sed '/GTID\_PURGED/d' dumpM1.sql > dumpM1\_nopurge.sql**

**sed '/GTID\_PURGED/d' dumpM2.sql > dumpM2\_nopurge.sql**

Use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to import each edited dump file into the replica. For example:

**mysql -u<*user*> -p<*password*> < dumpM1\_nopurge.sql**

**mysql -u<*user*> -p<*password*> < dumpM2\_nopurge.sql**

On the replica, issue [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) to clear the GTID execution history (assuming, as explained above, that all the dump files have been imported and that there are no wanted transactions with GTIDs on the replica). Then issue a **SET @@GLOBAL.gtid\_purged** statement to set the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) value to the union of all the GTID sets from all the dump files, as you recorded in Step 2. For example:

mysql> **RESET MASTER;**

mysql> **SET @@GLOBAL.gtid\_purged = "2174B383-5441-11E8-B90A-C80AA9429562:1-1029, 224DA167-0C0C-11E8-8442-00059A3C7B00:1-2695";**

If there are, or might be, overlapping transactions between the GTID sets in the dump files, you can use the stored functions described in [Section 17.1.3.8, “Stored Function Examples to Manipulate GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-functions) to check this beforehand and to calculate the union of all the GTID sets.

**17.1.5.3 Adding GTID-Based Sources to a Multi-Source Replica**

These steps assume you have enabled GTIDs for transactions on the sources using [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), created a replication user, ensured that the replica is using **TABLE** based replication applier metadata repositories, and provisioned the replica with data from the sources if appropriate.

Use the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) to configure a replication channel for each source on the replica (see [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels)). The **FOR CHANNEL** clause is used to specify the channel. For GTID-based replication, GTID auto-positioning is used to synchronize with the source (see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning)). The **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option is set to specify the use of auto-positioning.

For example, to add **source1** and **source2** as sources to the replica, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue the statement twice on the replica, like this:

mysql> **CHANGE MASTER TO MASTER\_HOST="source1", MASTER\_USER="ted", \**

**MASTER\_PASSWORD="*password*", MASTER\_AUTO\_POSITION=1 FOR CHANNEL "source\_1";**

mysql> **CHANGE MASTER TO MASTER\_HOST="source2", MASTER\_USER="ted", \**

**MASTER\_PASSWORD="*password*", MASTER\_AUTO\_POSITION=1 FOR CHANNEL "source\_2";**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_HOST="source1", SOURCE\_USER="ted", \**

**SOURCE\_PASSWORD="*password*", SOURCE\_AUTO\_POSITION=1 FOR CHANNEL "source\_1";**

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_HOST="source2", SOURCE\_USER="ted", \**

**SOURCE\_PASSWORD="*password*", SOURCE\_AUTO\_POSITION=1 FOR CHANNEL "source\_2";**

To make the replica replicate only database **db1** from **source1**, and only database **db2** from **source2**, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement for each channel, like this:

mysql> **CHANGE REPLICATION FILTER REPLICATE\_WILD\_DO\_TABLE = ('db1.%') FOR CHANNEL "source\_1";**

mysql> **CHANGE REPLICATION FILTER REPLICATE\_WILD\_DO\_TABLE = ('db2.%') FOR CHANNEL "source\_2";**

For the full syntax of the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement and other available options, see [Section 13.4.2.2, “CHANGE REPLICATION FILTER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

**17.1.5.4 Adding Binary Log Based Replication Sources to a Multi-Source Replica**

These steps assume that binary logging is enabled on the source (which is the default), the replica is using **TABLE** based replication applier metadata repositories (which is the default in MySQL 8.0), and that you have enabled a replication user and noted the current binary log file name and position.

Use the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) to configure a replication channel for each source on the replica (see [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels)). The **FOR CHANNEL** clause is used to specify the channel. For example, to add **source1** and **source2** as sources to the replica, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue the statement twice on the replica, like this:

mysql> **CHANGE MASTER TO MASTER\_HOST="source1", MASTER\_USER="ted", MASTER\_PASSWORD="*password*", \**

**MASTER\_LOG\_FILE='source1-bin.000006', MASTER\_LOG\_POS=628 FOR CHANNEL "source\_1";**

mysql> **CHANGE MASTER TO MASTER\_HOST="source2", MASTER\_USER="ted", MASTER\_PASSWORD="*password*", \**

**MASTER\_LOG\_FILE='source2-bin.000018', MASTER\_LOG\_POS=104 FOR CHANNEL "source\_2";**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_HOST="source1", SOURCE\_USER="ted", SOURCE\_PASSWORD="*password*", \**

**SOURCE\_LOG\_FILE='source1-bin.000006', SOURCE\_LOG\_POS=628 FOR CHANNEL "source\_1";**

mysql> **CHANGE REPLICATION SOURCE TO SOURCE\_HOST="source2", SOURCE\_USER="ted", SOURCE\_PASSWORD="*password*", \**

**SOURCE\_LOG\_FILE='source2-bin.000018', SOURCE\_LOG\_POS=104 FOR CHANNEL "source\_2";**

To make the replica replicate only database **db1** from **source1**, and only database **db2** from **source2**, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement for each channel, like this:

mysql> **CHANGE REPLICATION FILTER REPLICATE\_WILD\_DO\_TABLE = ('db1.%') FOR CHANNEL "source\_1";**

mysql> **CHANGE REPLICATION FILTER REPLICATE\_WILD\_DO\_TABLE = ('db2.%') FOR CHANNEL "source\_2";**

For the full syntax of the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement and other available options, see [Section 13.4.2.2, “CHANGE REPLICATION FILTER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

**17.1.5.5 Starting Multi-Source Replicas**

Once you have added channels for all of the replication sources, issue a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement to start replication. When you have enabled multiple channels on a replica, you can choose to either start all channels, or select a specific channel to start. For example, to start the two channels separately, use the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to issue the following statements:

mysql> **START SLAVE FOR CHANNEL "source\_1";**

mysql> **START SLAVE FOR CHANNEL "source\_2";**

Or from MySQL 8.0.22:

mysql> **START REPLICA FOR CHANNEL "source\_1";**

mysql> **START REPLICA FOR CHANNEL "source\_2";**

For the full syntax of the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) command and other available options, see [Section 13.4.2.7, “START REPLICA | SLAVE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica).

To verify that both channels have started and are operating correctly, you can issue [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statements on the replica, for example:

mysql> **SHOW SLAVE STATUS FOR CHANNEL "source\_1"\G**

mysql> **SHOW SLAVE STATUS FOR CHANNEL "source\_2"\G**

Or from MySQL 8.0.22:

mysql> **SHOW REPLICA STATUS FOR CHANNEL "source\_1"\G**

mysql> **SHOW REPLICA STATUS FOR CHANNEL "source\_2"\G**

**17.1.5.6 Stopping Multi-Source Replicas**

The [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement can be used to stop a multi-source replica. By default, if you use the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement on a multi-source replica all channels are stopped. Optionally, use the **FOR CHANNEL *channel*** clause to stop only a specific channel.

To stop all currently configured replication channels:

mysql> **STOP SLAVE;**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA;**

To stop only a named channel, use a **FOR CHANNEL *channel*** clause:

mysql> **STOP SLAVE FOR CHANNEL "source\_1";**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA FOR CHANNEL "source\_1";**

For the full syntax of the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) command and other available options, see [Section 13.4.2.9, “STOP REPLICA | SLAVE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica).

**17.1.5.7 Resetting Multi-Source Replicas**

The [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement can be used to reset a multi-source replica. By default, if you use the [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement on a multi-source replica all channels are reset. Optionally, use the **FOR CHANNEL *channel*** clause to reset only a specific channel.

To reset all currently configured replication channels:

mysql> **RESET SLAVE;**

Or from MySQL 8.0.22:

mysql> **RESET REPLICA;**

To reset only a named channel, use a **FOR CHANNEL *channel*** clause:

mysql> **RESET SLAVE FOR CHANNEL "source\_1";**

Or from MySQL 8.0.22:

mysql> **RESET REPLICA FOR CHANNEL "source\_1";**

For GTID-based replication, note that [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) has no effect on the replica's GTID execution history. If you want to clear this, issue [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) on the replica.

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) makes the replica forget its replication position, and clears the relay log, but it does not change any replication connection parameters (such as the source host name) or replication filters. If you want to remove these for a channel, issue [**RESET REPLICA | SLAVE ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica).

For the full syntax of the **RESET REPLICA | SLAVE** command and other available options, see [Section 13.4.2.5, “RESET REPLICA | SLAVE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica).

**17.1.5.8 Monitoring Multi-Source Replication**

To monitor the status of replication channels the following options exist:

Using the replication Performance Schema tables. The first column of these tables is **Channel\_Name**. This enables you to write complex queries based on **Channel\_Name** as a key. See [Section 27.12.11, “Performance Schema Replication Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-tables).

Using **SHOW REPLICA | SLAVE STATUS FOR CHANNEL *channel***. By default, if the **FOR CHANNEL *channel*** clause is not used, this statement shows the replica status for all channels with one row per channel. The identifier **Channel\_name** is added as a column in the result set. If a **FOR CHANNEL *channel*** clause is provided, the results show the status of only the named replication channel.

**Note**

The [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) statement does not work with multiple replication channels. The information that was available through these variables has been migrated to the replication performance tables. Using a [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) statement in a topology with multiple channels shows the status of only the default channel.

The error codes and messages that are issued when multi-source replication is enabled specify the channel that generated the error.

**17.1.5.8.1 Monitoring Channels Using Performance Schema Tables**

This section explains how to use the replication Performance Schema tables to monitor channels. You can choose to monitor all channels, or a subset of the existing channels.

To monitor the connection status of all channels:

mysql> **SELECT \* FROM replication\_connection\_status\G;**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CHANNEL\_NAME: source\_1

GROUP\_NAME:

SOURCE\_UUID: 046e41f8-a223-11e4-a975-0811960cc264

THREAD\_ID: 24

SERVICE\_STATE: ON

COUNT\_RECEIVED\_HEARTBEATS: 0

LAST\_HEARTBEAT\_TIMESTAMP: 0000-00-00 00:00:00

RECEIVED\_TRANSACTION\_SET: 046e41f8-a223-11e4-a975-0811960cc264:4-37

LAST\_ERROR\_NUMBER: 0

LAST\_ERROR\_MESSAGE:

LAST\_ERROR\_TIMESTAMP: 0000-00-00 00:00:00

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CHANNEL\_NAME: source\_2

GROUP\_NAME:

SOURCE\_UUID: 7475e474-a223-11e4-a978-0811960cc264

THREAD\_ID: 26

SERVICE\_STATE: ON

COUNT\_RECEIVED\_HEARTBEATS: 0

LAST\_HEARTBEAT\_TIMESTAMP: 0000-00-00 00:00:00

RECEIVED\_TRANSACTION\_SET: 7475e474-a223-11e4-a978-0811960cc264:4-6

LAST\_ERROR\_NUMBER: 0

LAST\_ERROR\_MESSAGE:

LAST\_ERROR\_TIMESTAMP: 0000-00-00 00:00:00

2 rows in set (0.00 sec)

In the above output there are two channels enabled, and as shown by the **CHANNEL\_NAME** field they are called **source\_1** and **source\_2**.

The addition of the **CHANNEL\_NAME** field enables you to query the Performance Schema tables for a specific channel. To monitor the connection status of a named channel, use a **WHERE CHANNEL\_NAME=*channel*** clause:

mysql> **SELECT \* FROM replication\_connection\_status WHERE CHANNEL\_NAME='source\_1'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CHANNEL\_NAME: source\_1

GROUP\_NAME:

SOURCE\_UUID: 046e41f8-a223-11e4-a975-0811960cc264

THREAD\_ID: 24

SERVICE\_STATE: ON

COUNT\_RECEIVED\_HEARTBEATS: 0

LAST\_HEARTBEAT\_TIMESTAMP: 0000-00-00 00:00:00

RECEIVED\_TRANSACTION\_SET: 046e41f8-a223-11e4-a975-0811960cc264:4-37

LAST\_ERROR\_NUMBER: 0

LAST\_ERROR\_MESSAGE:

LAST\_ERROR\_TIMESTAMP: 0000-00-00 00:00:00

1 row in set (0.00 sec)

Similarly, the **WHERE CHANNEL\_NAME=*channel*** clause can be used to monitor the other replication Performance Schema tables for a specific channel. For more information, see [Section 27.12.11, “Performance Schema Replication Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-tables).

**17.1.6 Replication and Binary Logging Options and Variables**

[17.1.6.1 Replication and Binary Logging Option and Variable Reference](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-reference)

[17.1.6.2 Replication Source Options and Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-source)

[17.1.6.3 Replica Server Options and Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-replica)

[17.1.6.4 Binary Logging Options and Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-binary-log)

[17.1.6.5 Global Transaction ID System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-gtids)

The following sections contain information about [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) options and server variables that are used in replication and for controlling the binary log. Options and variables for use on sources and replicas are covered separately, as are options and variables relating to binary logging and global transaction identifiers (GTIDs). A set of quick-reference tables providing basic information about these options and variables is also included.

Of particular importance is the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable.

|  |  |
| --- | --- |
| **Command-Line Format** | **--server-id=#** |
| **System Variable** | [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

This variable specifies the server ID. [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) is set to 1 by default. The server can be started with this default ID, but when binary logging is enabled, an informational message is issued if you did not set [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) explicitly to specify a server ID.

For servers that are used in a replication topology, you must specify a unique server ID for each replication server, in the range from 1 to 232 − 1. “Unique” means that each ID must be different from every other ID in use by any other source or replica in the replication topology. For additional information, see [Section 17.1.6.2, “Replication Source Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-source), and [Section 17.1.6.3, “Replica Server Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-replica).

If the server ID is set to 0, binary logging takes place, but a source with a server ID of 0 refuses any connections from replicas, and a replica with a server ID of 0 refuses to connect to a source. Note that although you can change the server ID dynamically to a nonzero value, doing so does not enable replication to start immediately. You must change the server ID and then restart the server to initialize the replica.

For more information, see [Section 17.1.2.2, “Setting the Replica Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto-slavebaseconfig).

**[server\_uuid](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_server_uuid)**

The MySQL server generates a true UUID in addition to the default or user-supplied server ID set in the **server\_id** system variable. This is available as the global, read-only variable [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid).

**Note**

The presence of the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) system variable does not change the requirement for setting a unique [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value for each MySQL server as part of preparing and running MySQL replication, as described earlier in this section.

|  |  |
| --- | --- |
| **System Variable** | [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

When starting, the MySQL server automatically obtains a UUID as follows:

Attempt to read and use the UUID written in the file ***data\_dir***/auto.cnf (where ***data\_dir*** is the server's data directory).

If ***data\_dir***/auto.cnf is not found, generate a new UUID and save it to this file, creating the file if necessary.

The auto.cnf file has a format similar to that used for my.cnf or my.ini files. auto.cnf has only a single **[auto]** section containing a single [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting and value; the file's contents appear similar to what is shown here:

[auto]

server\_uuid=8a94f357-aab4-11df-86ab-c80aa9429562

**Important**

The auto.cnf file is automatically generated; do not attempt to write or modify this file.

When using MySQL replication, sources and replicas know each other's UUIDs. The value of a replica's UUID can be seen in the output of [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas). Once [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) has been executed, the value of the source's UUID is available on the replica in the output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status).

**Note**

Issuing a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) or [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement does *not* reset the source's UUID as used on the replica.

A server's **server\_uuid** is also used in GTIDs for transactions originating on that server. For more information, see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids).

When starting, the replication I/O thread generates an error and aborts if its source's UUID is equal to its own unless the [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id) option has been set. In addition, the replication I/O thread generates a warning if either of the following is true:

No source having the expected [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) exists.

The source's [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) has changed, although no [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement has ever been executed.

**17.1.6.1 Replication and Binary Logging Option and Variable Reference**

The following two sections provide basic information about the MySQL command-line options and system variables applicable to replication and the binary log.

**Replication Options and Variables**

The command-line options and system variables in the following list relate to replication source servers and replicas. [Section 17.1.6.2, “Replication Source Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-source) provides more detailed information about options and variables relating to replication source servers. For more information about options and variables relating to replicas, see [Section 17.1.6.3, “Replica Server Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-replica).

**[abort-slave-event-count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_abort-slave-event-count)**: Option used by mysql-test for debugging and testing of replication.

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment): AUTO\_INCREMENT columns are incremented by this value.

[**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset): Offset added to AUTO\_INCREMENT columns.

[**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds): Purge binary logs after this many seconds.

[**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery): Controls how binary logs are iterated during GTID recovery.

[**Com\_change\_master**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of CHANGE MASTER TO statements.

[**Com\_show\_master\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW MASTER STATUS statements.

[**Com\_show\_slave\_hosts**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW REPLICAS and SHOW SLAVE HOSTS statements.

[**Com\_show\_replicas**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW REPLICAS and SHOW SLAVE HOSTS statements.

[**Com\_show\_slave\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW REPLICA STATUS and SHOW SLAVE STATUS statements.

[**Com\_show\_replica\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW REPLICA STATUS and SHOW SLAVE STATUS statements.

[**Com\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of START REPLICA and START SLAVE statements.

[**Com\_replica\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of START REPLICA and START SLAVE statements.

[**Com\_slave\_stop**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of STOP REPLICA and STOP SLAVE statements.

[**Com\_replica\_stop**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of STOP REPLICA and STOP SLAVE statements.

[**disconnect-slave-event-count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_disconnect-slave-event-count): Option used by mysql-test for debugging and testing of replication.

[**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency): Prevents execution of statements that cannot be logged in transactionally safe manner.

[**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days): Purge binary logs after this many days.

[**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed): Global: All GTIDs in binary log (global) or current transaction (session). Read-only.

[**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period): Compress gtid\_executed table each time this many transactions have occurred. 0 means never compress this table. Applies only when binary logging is disabled.

[**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode): Controls whether GTID based logging is enabled and what type of transactions logs can contain.

[**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next): Specifies GTID for next statement to execute; see documentation for details.

[**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned): Set of GTIDs owned by this client (session), or by all clients, together with thread ID of owner (global). Read-only.

[**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged): Set of all GTIDs that have been purged from binary log.

[**init\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_init_slave): Statements that are executed when replica connects to source.

[**log\_bin\_trust\_function\_creators**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_trust_function_creators): If equal to 0 (default), then when --log-bin is used, stored function creation is allowed only to users having SUPER privilege and only if function created does not break binary logging.

[**log\_statements\_unsafe\_for\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_statements_unsafe_for_binlog): Disables error 1592 warnings being written to error log.

[**master-info-file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-info-file): Location and name of file that remembers source and where I/O replication thread is in source's binary log.

[**master-retry-count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-retry-count): Number of tries replica makes to connect to source before giving up.

[**master\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository): Whether to write connection metadata repository, containing source information and replication I/O thread location in source's binary log, to file or table.

[**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size): If nonzero, relay log is rotated automatically when its size exceeds this value. If zero, size at which rotation occurs is determined by value of max\_binlog\_size.

[**original\_commit\_timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_commit_timestamp): Time when transaction was committed on original source.

[**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version): MySQL Server release number of server which is immediate replication source.

[**original\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_server_version): MySQL Server release number of server on which transaction was originally committed.

[**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log): Location and base name to use for relay logs.

[**relay\_log\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_basename): Complete path to relay log, including file name.

[**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index): Location and name to use for file that keeps list of last relay logs.

[**relay\_log\_info\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_file): File name for applier metadata repository in which replica records information about relay logs.

[**relay\_log\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository): Whether to write location of replication SQL thread in relay logs to file or table.

[**relay\_log\_purge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_purge): Determines whether relay logs are purged.

[**relay\_log\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery): Whether automatic recovery of relay log files from source at startup is enabled; must be enabled for crash-safe replica.

[**relay\_log\_space\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_space_limit): Maximum space to use for all relay logs.

[**replicate-do-db**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db): Tells replication SQL thread to restrict replication to specified database.

[**replicate-do-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table): Tells replication SQL thread to restrict replication to specified table.

[**replicate-ignore-db**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db): Tells replication SQL thread not to replicate to specified database.

[**replicate-ignore-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table): Tells replication SQL thread not to replicate to specified table.

[**replicate-rewrite-db**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db): Updates to database with different name from original.

[**replicate-same-server-id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id): In replication, if enabled, do not skip events having our server id.

[**replicate-wild-do-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table): Tells replication SQL thread to restrict replication to tables that match specified wildcard pattern.

[**replicate-wild-ignore-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table): Tells replication SQL thread not to replicate to tables that match given wildcard pattern.

[**report\_host**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host): Host name or IP of replica to be reported to source during replica registration.

[**report\_password**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_password): Arbitrary password which replica server should report to source; not same as password for replication user account.

[**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port): Port for connecting to replica reported to source during replica registration.

[**report\_user**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_user): Arbitrary user name which replica server should report to source; not same as name used for replication user account.

[**Rpl\_semi\_sync\_master\_clients**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_clients): Number of semisynchronous replicas.

[**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled): Whether semisynchronous replication is enabled on source.

[**Rpl\_semi\_sync\_master\_net\_avg\_wait\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_net_avg_wait_time): Average time source has waited for replies from replica.

[**Rpl\_semi\_sync\_master\_net\_wait\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_net_wait_time): Total time source has waited for replies from replica.

[**Rpl\_semi\_sync\_master\_net\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_net_waits): Total number of times source waited for replies from replica.

[**Rpl\_semi\_sync\_master\_no\_times**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_no_times): Number of times source turned off semisynchronous replication.

[**Rpl\_semi\_sync\_master\_no\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_no_tx): Number of commits not acknowledged successfully.

[**Rpl\_semi\_sync\_master\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_status): Whether semisynchronous replication is operational on source.

[**Rpl\_semi\_sync\_master\_timefunc\_failures**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_timefunc_failures): Number of times source failed when calling time functions.

[**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout): Number of milliseconds to wait for replica acknowledgment.

[**rpl\_semi\_sync\_master\_trace\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_trace_level): Semisynchronous replication debug trace level on source.

[**Rpl\_semi\_sync\_master\_tx\_avg\_wait\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_tx_avg_wait_time): Average time source waited for each transaction.

[**Rpl\_semi\_sync\_master\_tx\_wait\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_tx_wait_time): Total time source waited for transactions.

[**Rpl\_semi\_sync\_master\_tx\_waits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_tx_waits): Total number of times source waited for transactions.

[**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count): Number of replica acknowledgments source must receive per transaction before proceeding.

[**rpl\_semi\_sync\_master\_wait\_no\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_no_slave): Whether source waits for timeout even with no replicas.

[**rpl\_semi\_sync\_master\_wait\_point**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_point): Wait point for replica transaction receipt acknowledgment.

[**Rpl\_semi\_sync\_master\_wait\_pos\_backtraverse**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_wait_pos_backtraverse): Total number of times source has waited for event with binary coordinates lower than events waited for previously.

[**Rpl\_semi\_sync\_master\_wait\_sessions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_wait_sessions): Number of sessions currently waiting for replica replies.

[**Rpl\_semi\_sync\_master\_yes\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_yes_tx): Number of commits acknowledged successfully.

[**replication\_sender\_observe\_commit\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_sender_observe_commit_only): Limited callbacks for semisynchronous replication.

[**replication\_optimize\_for\_static\_plugin\_config**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_optimize_for_static_plugin_config): Shared locks for semisynchronous replication.

[**rpl\_semi\_sync\_slave\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_enabled): Whether semisynchronous replication is enabled on replica.

[**Rpl\_semi\_sync\_slave\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_slave_status): Whether semisynchronous replication is operational on replica.

[**rpl\_semi\_sync\_slave\_trace\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_trace_level): Semisynchronous replication debug trace level on replica.

[**rpl\_read\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_read_size): Set minimum amount of data in bytes which is read from binary log files and relay log files.

[**rpl\_stop\_slave\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_stop_slave_timeout): Number of seconds that STOP REPLICA or STOP SLAVE waits before timing out.

[**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid): Server's globally unique ID, automatically (re)generated at server start.

[**show-slave-auth-info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_show-slave-auth-info): Show user name and password in SHOW REPLICAS and SHOW SLAVE HOSTS on this source.

[**skip-slave-start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start): If set, replication is not autostarted when replica server starts.

[**slave\_load\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_load_tmpdir): Location where replica should put its temporary files when replicating LOAD DATA statements.

[**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout): Number of seconds to wait for more data from source/replica connection before aborting read.

[**slave-skip-errors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_slave-skip-errors): Tells replication thread to continue replication when query returns error from provided list.

[**slave\_checkpoint\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_group): Maximum number of transactions processed by multithreaded replica before checkpoint operation is called to update progress status. Not supported by NDB Cluster.

[**slave\_checkpoint\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_period): Update progress status of multithreaded replica and flush relay log info to disk after this number of milliseconds. Not supported by NDB Cluster.

[**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol): Use compression of source/replica protocol.

[**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode): Allows for switching replication thread between IDEMPOTENT mode (key and some other errors suppressed) and STRICT mode; STRICT mode is default, except for NDB Cluster, where IDEMPOTENT is always used.

[**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet): Maximum size, in bytes, of packet that can be sent from replication source server to replica; overrides max\_allowed\_packet.

[**Slave\_open\_temp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_open_temp_tables): Number of temporary tables that replication SQL thread currently has open.

[**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type): Tells replica to use timestamp information (LOGICAL\_CLOCK) or database partioning (DATABASE) to parallelize transactions.

[**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers): Number of applier threads for executing replication transactions in parallel. 0 disables replica multithreading. Not supported by MySQL Cluster.

[**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max): Maximum size of replica worker queues holding events not yet applied.

[**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order): Ensures that all commits by replica workers happen in same order as on source to maintain consistency when using parallel applier threads.

[**slave\_rows\_search\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_rows_search_algorithms): Determines search algorithms used for replica update batching. Any 2 or 3 from this list: INDEX\_SEARCH, TABLE\_SCAN, HASH\_SCAN.

[**Slave\_rows\_last\_search\_algorithm\_used**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_rows_last_search_algorithm_used): Search algorithm most recently used by this replica to locate rows for row-based replication (index, table, or hash scan).

[**slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries): Number of times replication SQL thread retries transaction in case it failed with deadlock or elapsed lock wait timeout, before giving up and stopping.

[**slave\_type\_conversions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_type_conversions): Controls type conversion mode on replica. Value is list of zero or more elements from this list: ALL\_LOSSY, ALL\_NON\_LOSSY. Set to empty string to disallow type conversions between source and replica.

[**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin): Controls binary logging for current session.

[**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter): Number of events from source that replica should skip. Not compatible with GTID replication.

[**sync\_master\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_master_info): Synchronize master.info to disk after every #th event.

[**sync\_relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log): Synchronize relay log to disk after every #th event.

[**sync\_relay\_log\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log_info): Synchronize relay.info file to disk after every #th event.

[**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction): Defines algorithm used to hash writes extracted during transaction.

For a listing of all command-line options, system variables, and status variables used with [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), see [Section 5.1.4, “Server Option, System Variable, and Status Variable Reference”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-option-variable-reference).

**Binary Logging Options and Variables**

The command-line options and system variables in the following list relate to the binary log. [Section 17.1.6.4, “Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-binary-log), provides more detailed information about options and variables relating to binary logging. For additional general information about the binary log, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

**[binlog-checksum](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-checksum)**: Enable/disable binary log checksums.

[**binlog-do-db**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db): Limits binary logging to specific databases.

[**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format): Specifies format of binary log.

[**binlog-ignore-db**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db): Tells source that updates to given database should not be written to binary log.

[**binlog-row-event-max-size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-row-event-max-size): Binary log max event size.

[**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption): Enable encryption for binary log files and relay log files on this server.

[**binlog\_rotate\_encryption\_master\_key\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rotate_encryption_master_key_at_startup): Rotate binary log master key at server startup.

[**Binlog\_cache\_disk\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_cache_disk_use): Number of transactions which used temporary file instead of binary log cache.

[**binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_cache_size): Size of cache to hold SQL statements for binary log during transaction.

[**Binlog\_cache\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_cache_use): Number of transactions that used temporary binary log cache.

[**binlog\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum): Enable/disable binary log checksums.

[**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates): Causes updates using statement format to nontransactional engines to be written directly to binary log. See documentation before using.

[**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action): Controls what happens when server cannot write to binary log.

[**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay): Sets number of microseconds to wait before synchronizing transactions to disk.

[**binlog\_group\_commit\_sync\_no\_delay\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_no_delay_count): Sets maximum number of transactions to wait for before aborting current delay specified by binlog\_group\_commit\_sync\_delay.

[**binlog\_max\_flush\_queue\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_max_flush_queue_time): How long to read transactions before flushing to binary log.

[**binlog\_order\_commits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_order_commits): Whether to commit in same order as writes to binary log.

[**binlog\_row\_image**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_image): Use full or minimal images when logging row changes.

[**binlog\_row\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_metadata): Whether to record all or only minimal table related metadata to binary log when using row-based logging.

[**binlog\_row\_value\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_value_options): Enables binary logging of partial JSON updates for row-based replication.

[**binlog\_rows\_query\_log\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rows_query_log_events): When enabled, enables logging of rows query log events when using row-based logging. Disabled by default. Do not enable when producing logs for pre-5.6 replicas/readers.

[**Binlog\_stmt\_cache\_disk\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_stmt_cache_disk_use): Number of nontransactional statements that used temporary file instead of binary log statement cache.

[**binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_stmt_cache_size): Size of cache to hold nontransactional statements for binary log during transaction.

[**Binlog\_stmt\_cache\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_stmt_cache_use): Number of statements that used temporary binary log statement cache.

[**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression): Enable compression for transaction payloads in binary log files.

[**binlog\_transaction\_compression\_level\_zstd**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression_level_zstd): Compression level for transaction payloads in binary log files.

[**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking): Source of dependency information (commit timestamps or transaction write sets) from which to assess which transactions can be executed in parallel by replica's multithreaded applier.

[**binlog\_transaction\_dependency\_history\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_history_size): Number of row hashes kept for looking up transaction that last updated some row.

[**Com\_show\_binlog\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW BINLOG EVENTS statements.

[**Com\_show\_binlogs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): Count of SHOW BINLOGS statements.

[**log-bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin): Base name for binary log files.

[**log-bin-index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin-index): Name of binary log index file.

[**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin): Whether binary log is enabled.

[**log\_bin\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_basename): Path and base name for binary log files.

[**log\_bin\_use\_v1\_row\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_use_v1_row_events): Whether server is using version 1 binary log row events.

[**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates): Whether replica should log updates performed by its replication SQL thread to its own binary log.

[**master\_verify\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_verify_checksum): Cause source to examine checksums when reading from binary log.

[**max-binlog-dump-events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_max-binlog-dump-events): Option used by mysql-test for debugging and testing of replication.

[**max\_binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_cache_size): Can be used to restrict total size used to cache multi-statement transaction.

[**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size): Binary log is rotated automatically when size exceeds this value.

[**max\_binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_stmt_cache_size): Can be used to restrict total size used to cache all nontransactional statements during transaction.

[**slave-sql-verify-checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_slave-sql-verify-checksum): Cause replica to examine checksums when reading from relay log.

[**slave\_sql\_verify\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_sql_verify_checksum): Cause replica to examine checksums when reading from relay log.

[**sporadic-binlog-dump-fail**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_sporadic-binlog-dump-fail): Option used by mysql-test for debugging and testing of replication.

[**sync\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog): Synchronously flush binary log to disk after every #th event.

For a listing of all command-line options, system and status variables used with [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), see [Section 5.1.4, “Server Option, System Variable, and Status Variable Reference”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-option-variable-reference).

**17.1.6.2 Replication Source Options and Variables**

This section describes the server options and system variables that you can use on replication source servers. You can specify the options either on the [command line](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#command-line-options) or in an [option file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files). You can specify system variable values using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

On the source and each replica, you must set the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable to establish a unique replication ID. For each server, you should pick a unique positive integer in the range from 1 to 232 − 1, and each ID must be different from every other ID in use by any other source or replica in the replication topology. Example: **server-id=3**.

For options used on the source for controlling binary logging, see [Section 17.1.6.4, “Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-binary-log).

**Startup Options for Replication Source Servers**

The following list describes startup options for controlling replication source servers. Replication-related system variables are discussed later in this section.

[--show-slave-auth-info](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_show-slave-auth-info)

|  |  |
| --- | --- |
| **Command-Line Format** | **--show-slave-auth-info[={OFF|ON}]** |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Display replication user names and passwords in the output of [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) on the source for replicas started with the [--report-user](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_user) and [--report-password](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_password) options.

**System Variables Used on Replication Source Servers**

The following system variables are used for or by replication source servers:

**[auto\_increment\_increment](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_auto_increment_increment)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--auto-increment-increment=#** |
| **System Variable** | [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | Yes |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **65535** |

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) are intended for use with circular (source-to-source) replication, and can be used to control the operation of **AUTO\_INCREMENT** columns. Both variables have global and session values, and each can assume an integer value between 1 and 65,535 inclusive. Setting the value of either of these two variables to 0 causes its value to be set to 1 instead. Attempting to set the value of either of these two variables to an integer greater than 65,535 or less than 0 causes its value to be set to 65,535 instead. Attempting to set the value of [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) or [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) to a noninteger value produces an error, and the actual value of the variable remains unchanged.

**Note**

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) is also supported for use with [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables.

As of MySQL 8.0.18, setting the session value of this system variable is no longer a restricted operation.

When Group Replication is started on a server, the value of [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) is changed to the value of [**group\_replication\_auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_auto_increment_increment), which defaults to 7, and the value of [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) is changed to the server ID. The changes are reverted when Group Replication is stopped. These changes are only made and reverted if [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) each have their default value of 1. If their values have already been modified from the default, Group Replication does not alter them. From MySQL 8.0, the system variables are also not modified when Group Replication is in single-primary mode, where only one server writes.

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) affect **AUTO\_INCREMENT** column behavior as follows:

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) controls the interval between successive column values. For example:

mysql> **SHOW VARIABLES LIKE 'auto\_inc%';**

+--------------------------+-------+

| Variable\_name | Value |

+--------------------------+-------+

| auto\_increment\_increment | 1 |

| auto\_increment\_offset | 1 |

+--------------------------+-------+

2 rows in set (0.00 sec)

mysql> **CREATE TABLE autoinc1**

-> **(col INT NOT NULL AUTO\_INCREMENT PRIMARY KEY);**

Query OK, 0 rows affected (0.04 sec)

mysql> **SET @@auto\_increment\_increment=10;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SHOW VARIABLES LIKE 'auto\_inc%';**

+--------------------------+-------+

| Variable\_name | Value |

+--------------------------+-------+

| auto\_increment\_increment | 10 |

| auto\_increment\_offset | 1 |

+--------------------------+-------+

2 rows in set (0.01 sec)

mysql> **INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);**

Query OK, 4 rows affected (0.00 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> **SELECT col FROM autoinc1;**

+-----+

| col |

+-----+

| 1 |

| 11 |

| 21 |

| 31 |

+-----+

4 rows in set (0.00 sec)

[**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) determines the starting point for the **AUTO\_INCREMENT** column value. Consider the following, assuming that these statements are executed during the same session as the example given in the description for [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment):

mysql> **SET @@auto\_increment\_offset=5;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SHOW VARIABLES LIKE 'auto\_inc%';**

+--------------------------+-------+

| Variable\_name | Value |

+--------------------------+-------+

| auto\_increment\_increment | 10 |

| auto\_increment\_offset | 5 |

+--------------------------+-------+

2 rows in set (0.00 sec)

mysql> **CREATE TABLE autoinc2**

-> **(col INT NOT NULL AUTO\_INCREMENT PRIMARY KEY);**

Query OK, 0 rows affected (0.06 sec)

mysql> **INSERT INTO autoinc2 VALUES (NULL), (NULL), (NULL), (NULL);**

Query OK, 4 rows affected (0.00 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> **SELECT col FROM autoinc2;**

+-----+

| col |

+-----+

| 5 |

| 15 |

| 25 |

| 35 |

+-----+

4 rows in set (0.02 sec)

When the value of [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) is greater than that of [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment), the value of [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) is ignored.

If either of these variables is changed, and then new rows inserted into a table containing an **AUTO\_INCREMENT** column, the results may seem counterintuitive because the series of **AUTO\_INCREMENT** values is calculated without regard to any values already present in the column, and the next value inserted is the least value in the series that is greater than the maximum existing value in the **AUTO\_INCREMENT** column. The series is calculated like this:

**auto\_increment\_offset** + ***N*** × **auto\_increment\_increment**

where ***N*** is a positive integer value in the series [1, 2, 3, ...]. For example:

mysql> **SHOW VARIABLES LIKE 'auto\_inc%';**

+--------------------------+-------+

| Variable\_name | Value |

+--------------------------+-------+

| auto\_increment\_increment | 10 |

| auto\_increment\_offset | 5 |

+--------------------------+-------+

2 rows in set (0.00 sec)

mysql> **SELECT col FROM autoinc1;**

+-----+

| col |

+-----+

| 1 |

| 11 |

| 21 |

| 31 |

+-----+

4 rows in set (0.00 sec)

mysql> **INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);**

Query OK, 4 rows affected (0.00 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> **SELECT col FROM autoinc1;**

+-----+

| col |

+-----+

| 1 |

| 11 |

| 21 |

| 31 |

| 35 |

| 45 |

| 55 |

| 65 |

+-----+

8 rows in set (0.00 sec)

The values shown for [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) and [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) generate the series 5 + ***N*** × 10, that is, [5, 15, 25, 35, 45, ...]. The highest value present in the **col** column prior to the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is 31, and the next available value in the **AUTO\_INCREMENT** series is 35, so the inserted values for **col** begin at that point and the results are as shown for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) query.

It is not possible to restrict the effects of these two variables to a single table; these variables control the behavior of all **AUTO\_INCREMENT** columns in *all* tables on the MySQL server. If the global value of either variable is set, its effects persist until the global value is changed or overridden by setting the session value, or until [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is restarted. If the local value is set, the new value affects **AUTO\_INCREMENT** columns for all tables into which new rows are inserted by the current user for the duration of the session, unless the values are changed during that session.

The default value of [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment) is 1. See [Section 17.5.1.1, “Replication and AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-auto-increment).

**[auto\_increment\_offset](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_auto_increment_offset)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--auto-increment-offset=#** |
| **System Variable** | [**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | Yes |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **65535** |

This variable has a default value of 1. If it is left with its default value, and Group Replication is started on the server in multi-primary mode, it is changed to the server ID. For more information, see the description for [**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment).

**Note**

**auto\_increment\_offset** is also supported for use with [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables.

As of MySQL 8.0.18, setting the session value of this system variable is no longer a restricted operation.

**[immediate\_server\_version](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_immediate_server_version)**

|  |  |
| --- | --- |
| **Introduced** | 8.0.14 |
| **System Variable** | [**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |

For internal use by replication. This session system variable holds the MySQL Server release number of the server that is the immediate source in a replication topology (for example, **80014** for a MySQL 8.0.14 server instance). If this immediate server is at a release that does not support the session system variable, the value of the variable is set to 0 (**UNKNOWN\_SERVER\_VERSION**).

The value of the variable is replicated from a source to a replica. With this information the replica can correctly process data originating from a source at an older release, by recognizing where syntax changes or semantic changes have occurred between the releases involved and handling these appropriately. The information can also be used in a Group Replication environment where one or more members of the replication group is at a newer release than the others. The value of the variable can be viewed in the binary log for each transaction (as part of the **Gtid\_log\_event**, or **Anonymous\_gtid\_log\_event** if GTIDs are not in use on the server), and could be helpful in debugging cross-version replication issues.

Setting the session value of this system variable is a restricted operation. The session user must have either the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege (see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)), or privileges sufficient to set restricted session variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)). However, note that the variable is not intended for users to set; it is set automatically by the replication infrastructure.

**[original\_server\_version](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_original_server_version)**

|  |  |
| --- | --- |
| **Introduced** | 8.0.14 |
| **System Variable** | [**original\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_server_version) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |

For internal use by replication. This session system variable holds the MySQL Server release number of the server where a transaction was originally committed (for example, **80014** for a MySQL 8.0.14 server instance). If this original server is at a release that does not support the session system variable, the value of the variable is set to 0 (**UNKNOWN\_SERVER\_VERSION**). Note that when a release number is set by the original server, the value of the variable is reset to 0 if the immediate server or any other intervening server in the replication topology does not support the session system variable, and so does not replicate its value.

The value of the variable is set and used in the same ways as for the [**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version) system variable. If the value of the variable is the same as that for the [**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version) system variable, only the latter is recorded in the binary log, with an indicator that the original server version is the same.

In a Group Replication environment, view change log events, which are special transactions queued by each group member when a new member joins the group, are tagged with the server version of the group member queuing the transaction. This ensures that the server version of the original donor is known to the joining member. Because the view change log events queued for a particular view change have the same GTID on all members, for this case only, instances of the same GTID might have a different original server version.

Setting the session value of this system variable is a restricted operation. The session user must have either the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege (see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)), or privileges sufficient to set restricted session variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)). However, note that the variable is not intended for users to set; it is set automatically by the replication infrastructure.

**[rpl\_semi\_sync\_master\_enabled](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_enabled)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-enabled[={OFF|ON}]** |
| **System Variable** | [**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Controls whether semisynchronous replication is enabled on the source server. To enable or disable the plugin, set this variable to **ON** or **OFF** (or 1 or 0), respectively. The default is **OFF**.

This variable is available only if the source-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_master\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-timeout=#** |
| **System Variable** | [**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10000** |

A value in milliseconds that controls how long the source waits on a commit for acknowledgment from a replica before timing out and reverting to asynchronous replication. The default value is 10000 (10 seconds).

This variable is available only if the source-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_master\_trace\_level](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_trace_level)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-trace-level=#** |
| **System Variable** | [**rpl\_semi\_sync\_master\_trace\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_trace_level) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **32** |

The semisynchronous replication debug trace level on the source server. Four levels are defined:

1 = general level (for example, time function failures)

16 = detail level (more verbose information)

32 = net wait level (more information about network waits)

64 = function level (information about function entry and exit)

This variable is available only if the source-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_master\_wait\_for\_slave\_count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_wait_for_slave_count)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-wait-for-slave-count=#** |
| **System Variable** | [**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **1** |
| **Maximum Value** | **65535** |

The number of replica acknowledgments the source must receive per transaction before proceeding. By default **rpl\_semi\_sync\_master\_wait\_for\_slave\_count** is **1**, meaning that semisynchronous replication proceeds after receiving a single replica acknowledgment. Performance is best for small values of this variable.

For example, if **rpl\_semi\_sync\_master\_wait\_for\_slave\_count** is **2**, then 2 replicas must acknowledge receipt of the transaction before the timeout period configured by [**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout) for semisynchronous replication to proceed. If fewer replicas acknowledge receipt of the transaction during the timeout period, the source reverts to normal replication.

**Note**

This behavior also depends on [**rpl\_semi\_sync\_master\_wait\_no\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_no_slave)

This variable is available only if the source-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_master\_wait\_no\_slave](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_wait_no_slave)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-wait-no-slave[={OFF|ON}]** |
| **System Variable** | [**rpl\_semi\_sync\_master\_wait\_no\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_no_slave) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Controls whether the source waits for the timeout period configured by [**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout) to expire, even if the replica count drops to less than the number of replicas configured by [**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count) during the timeout period.

When the value of **rpl\_semi\_sync\_master\_wait\_no\_slave** is **ON** (the default), it is permissible for the replica count to drop to less than [**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count) during the timeout period. As long as enough replicas acknowledge the transaction before the timeout period expires, semisynchronous replication continues.

When the value of **rpl\_semi\_sync\_master\_wait\_no\_slave** is **OFF**, if the replica count drops to less than the number configured in [**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count) at any time during the timeout period configured by [**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout), the source reverts to normal replication.

This variable is available only if the source-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_master\_wait\_point](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_master_wait_point)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-master-wait-point=value** |
| **System Variable** | [**rpl\_semi\_sync\_master\_wait\_point**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_point) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **AFTER\_SYNC** |
| **Valid Values** | **AFTER\_SYNC**  **AFTER\_COMMIT** |

This variable controls the point at which a semisynchronous replication source server waits for replica acknowledgment of transaction receipt before returning a status to the client that committed the transaction. These values are permitted:

**AFTER\_SYNC** (the default): The source writes each transaction to its binary log and the replica, and syncs the binary log to disk. The source waits for replica acknowledgment of transaction receipt after the sync. Upon receiving acknowledgment, the source commits the transaction to the storage engine and returns a result to the client, which then can proceed.

**AFTER\_COMMIT**: The source writes each transaction to its binary log and the replica, syncs the binary log, and commits the transaction to the storage engine. The source waits for replica acknowledgment of transaction receipt after the commit. Upon receiving acknowledgment, the source returns a result to the client, which then can proceed.

The replication characteristics of these settings differ as follows:

With **AFTER\_SYNC**, all clients see the committed transaction at the same time: After it has been acknowledged by the replica and committed to the storage engine on the source. Thus, all clients see the same data on the source.

In the event of source failure, all transactions committed on the source have been replicated to the replica (saved to its relay log). An unexpected exit of the source server and failover to the replica is lossless because the replica is up to date. Note, however, that the source cannot be restarted in this scenario and must be discarded, because its binary log might contain uncommitted transactions that would cause a conflict with the replica when externalized after binary log recovery.

With **AFTER\_COMMIT**, the client issuing the transaction gets a return status only after the server commits to the storage engine and receives replica acknowledgment. After the commit and before replica acknowledgment, other clients can see the committed transaction before the committing client.

If something goes wrong such that the replica does not process the transaction, then in the event of an unexpected source server exit and failover to the replica, it is possible for such clients to see a loss of data relative to what they saw on the source.

This variable is available only if the source-side semisynchronous replication plugin is installed.

With the addition of [**rpl\_semi\_sync\_master\_wait\_point**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_point) in MySQL 5.7, a version compatibility constraint was created because it increments the semisynchronous interface version: Servers for MySQL 5.7 and higher do not work with semisynchronous replication plugins from older versions, nor do servers from older versions work with semisynchronous replication plugins for MySQL 5.7 and higher.

**17.1.6.3 Replica Server Options and Variables**

This section explains the server options and system variables that apply to replica servers and contains the following:

[Startup Options for Replica Servers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-optvars-slaves)

[System Variables Used on Replica Servers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sysvars-slaves)

Specify the options either on the [command line](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#command-line-options) or in an [option file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option-files). Many of the options can be set while the server is running by using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). Specify system variable values using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

**Server ID.** On the source and each replica, you must set the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable to establish a unique replication ID in the range from 1 to 232 − 1. “Unique” means that each ID must be different from every other ID in use by any other source or replica in the replication topology. Example my.cnf file:

[mysqld]

server-id=3

**Startup Options for Replica Servers**

This section explains startup options for controlling replica servers. Many of these options can be set while the server is running by using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). Others, such as the --replicate-\* options, can be set only when the replica server starts. Replication-related system variables are discussed later in this section.

[--master-info-file=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_master-info-file)***[file\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_master-info-file)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--master-info-file=file\_name** |
| **Deprecated** | 8.0.18 |
| **Type** | File name |
| **Default Value** | **master.info** |

The use of this option is now deprecated. It was used to set the file name for the replica's connection metadata repository if [**master\_info\_repository=FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) was set. [--master-info-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-info-file) and the use of the [**master\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) system variable are deprecated because the use of a file for the connection metadata repository has been superseded by crash-safe tables. For information about the connection metadata repository, see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status).

[--master-retry-count=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_master-retry-count)***[count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_master-retry-count)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--master-retry-count=#** |
| **Deprecated** | Yes |
| **Type** | Integer |
| **Default Value** | **86400** |
| **Minimum Value** | **0** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

The number of times that the replica tries to reconnect to the source before giving up. The default value is 86400 times. A value of 0 means “infinite”, and the replica attempts to connect forever. Reconnection attempts are triggered when the replica reaches its connection timeout (specified by the [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable) without receiving data or a heartbeat signal from the source. Reconnection is attempted at intervals set by the **SOURCE\_CONNECT\_RETRY** | **MASTER\_CONNECT\_RETRY** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (which defaults to every 60 seconds).

This option is deprecated; expect it to be removed in a future MySQL release. Use the **SOURCE\_RETRY\_COUNT** | **MASTER\_RETRY\_COUNT** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement instead.

[--max-relay-log-size=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_max-relay-log-size)***[size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_max-relay-log-size)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-relay-log-size=#** |
| **System Variable** | [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1073741824** |

The size at which the server rotates relay log files automatically. If this value is nonzero, the relay log is rotated automatically when its size exceeds this value. If this value is zero (the default), the size at which relay log rotation occurs is determined by the value of [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size). For more information, see [Section 17.2.4.1, “The Relay Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-relaylog).

[--relay-log-purge={0|1}](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_relay-log-purge)

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-purge[={OFF|ON}]** |
| **System Variable** | [**relay\_log\_purge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_purge) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Disable or enable automatic purging of relay logs as soon as they are no longer needed. The default value is 1 (enabled). This is a global variable that can be changed dynamically with **SET GLOBAL relay\_log\_purge = *N***. Disabling purging of relay logs when enabling the [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) option risks data consistency and is therefore not crash-safe.

[--relay-log-space-limit=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_relay-log-space-limit)***[size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_relay-log-space-limit)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-space-limit=#** |
| **System Variable** | [**relay\_log\_space\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_space_limit) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

This option places an upper limit on the total size in bytes of all relay logs on the replica. A value of 0 means “no limit”. This is useful for a replica server host that has limited disk space. When the limit is reached, the I/O thread stops reading binary log events from the source server until the SQL thread has caught up and deleted some unused relay logs. Note that this limit is not absolute: There are cases where the SQL thread needs more events before it can delete relay logs. In that case, the I/O thread exceeds the limit until it becomes possible for the SQL thread to delete some relay logs because not doing so would cause a deadlock. You should not set [--relay-log-space-limit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_relay-log-space-limit) to less than twice the value of [--max-relay-log-size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_max-relay-log-size) (or [--max-binlog-size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) if [--max-relay-log-size](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_max-relay-log-size) is 0). In that case, there is a chance that the I/O thread waits for free space because [--relay-log-space-limit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_relay-log-space-limit) is exceeded, but the SQL thread has no relay log to purge and is unable to satisfy the I/O thread. This forces the I/O thread to ignore [--relay-log-space-limit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_relay-log-space-limit) temporarily.

[--replicate-do-db=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-do-db)***[db\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-do-db)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-do-db=name** |
| **Type** | String |

Creates a replication filter using the name of a database. Such filters can also be created using [**CHANGE REPLICATION FILTER REPLICATE\_DO\_DB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-do-db:***channel\_1***:***db\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

The precise effect of this replication filter depends on whether statement-based or row-based replication is in use.

**Statement-based replication.** Tell the replication SQL thread to restrict replication to statements where the default database (that is, the one selected by [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use)) is ***db\_name***. To specify more than one database, use this option multiple times, once for each database; however, doing so does *not* replicate cross-database statements such as **UPDATE *some\_db.some\_table* SET foo='bar'** while a different database (or no database) is selected.

**Warning**

To specify multiple databases you *must* use multiple instances of this option. Because database names can contain commas, if you supply a comma separated list then the list is treated as the name of a single database.

An example of what does not work as you might expect when using statement-based replication: If the replica is started with [--replicate-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) and you issue the following statements on the source, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement is *not* replicated:

USE prices;

UPDATE sales.january SET amount=amount+1000;

The main reason for this “check just the default database” behavior is that it is difficult from the statement alone to know whether it should be replicated (for example, if you are using multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements or multiple-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements that act across multiple databases). It is also faster to check only the default database rather than all databases if there is no need.

**Row-based replication.** Tells the replication SQL thread to restrict replication to database ***db\_name***. Only tables belonging to ***db\_name*** are changed; the current database has no effect on this. Suppose that the replica is started with [--replicate-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) and row-based replication is in effect, and then the following statements are run on the source:

USE prices;

UPDATE sales.february SET amount=amount+100;

The **february** table in the **sales** database on the replica is changed in accordance with the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement; this occurs whether or not the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement was issued. However, issuing the following statements on the source has no effect on the replica when using row-based replication and [--replicate-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db):

USE prices;

UPDATE prices.march SET amount=amount-25;

Even if the statement **USE prices** were changed to **USE sales**, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement's effects would still not be replicated.

Another important difference in how [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) is handled in statement-based replication as opposed to row-based replication occurs with regard to statements that refer to multiple databases. Suppose that the replica is started with [--replicate-do-db=db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db), and the following statements are executed on the source:

USE db1;

UPDATE db1.table1, db2.table2 SET db1.table1.col1 = 10, db2.table2.col2 = 20;

If you are using statement-based replication, then both tables are updated on the replica. However, when using row-based replication, only **table1** is affected on the replica; since **table2** is in a different database, **table2** on the replica is not changed by the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). Now suppose that, instead of the **USE db1** statement, a **USE db4** statement had been used:

USE db4;

UPDATE db1.table1, db2.table2 SET db1.table1.col1 = 10, db2.table2.col2 = 20;

In this case, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement would have no effect on the replica when using statement-based replication. However, if you are using row-based replication, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) would change **table1** on the replica, but not **table2**—in other words, only tables in the database named by [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) are changed, and the choice of default database has no effect on this behavior.

If you need cross-database updates to work, use [--replicate-wild-do-table=***db\_name***.%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) instead. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules).

**Note**

This option affects replication in the same manner that [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) affects binary logging, and the effects of the replication format on how [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) affects replication behavior are the same as those of the logging format on the behavior of [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db).

This option has no effect on [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statements.

[--replicate-ignore-db=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-ignore-db)***[db\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-ignore-db)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-ignore-db=name** |
| **Type** | String |

Creates a replication filter using the name of a database. Such filters can also be created using [**CHANGE REPLICATION FILTER REPLICATE\_IGNORE\_DB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-ignore-db:***channel\_1***:***db\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

To specify more than one database to ignore, use this option multiple times, once for each database. Because database names can contain commas, if you supply a comma-separated list, it is treated as the name of a single database.

As with [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db), the precise effect of this filtering depends on whether statement-based or row-based replication is in use, and are described in the next several paragraphs.

**Statement-based replication.** Tells the replication SQL thread not to replicate any statement where the default database (that is, the one selected by [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use)) is ***db\_name***.

**Row-based replication.** Tells the replication SQL thread not to update any tables in the database ***db\_name***. The default database has no effect.

When using statement-based replication, the following example does not work as you might expect. Suppose that the replica is started with [--replicate-ignore-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) and you issue the following statements on the source:

USE prices;

UPDATE sales.january SET amount=amount+1000;

The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement *is* replicated in such a case because [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) applies only to the default database (determined by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement). Because the **sales** database was specified explicitly in the statement, the statement has not been filtered. However, when using row-based replication, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement's effects are *not* propagated to the replica, and the replica's copy of the **sales.january** table is unchanged; in this instance, [--replicate-ignore-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) causes *all* changes made to tables in the source's copy of the **sales** database to be ignored by the replica.

You should not use this option if you are using cross-database updates and you do not want these updates to be replicated. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules).

If you need cross-database updates to work, use [--replicate-wild-ignore-table=***db\_name***.%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) instead. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules).

**Note**

This option affects replication in the same manner that [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) affects binary logging, and the effects of the replication format on how [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) affects replication behavior are the same as those of the logging format on the behavior of [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db).

This option has no effect on [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), or [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statements.

[--replicate-do-table=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-do-table)***[db\_name.tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-do-table)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-do-table=name** |
| **Type** | String |

Creates a replication filter by telling the replication SQL thread to restrict replication to a given table. To specify more than one table, use this option multiple times, once for each table. This works for both cross-database updates and default database updates, in contrast to [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db). See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules). You can also create such a filter by issuing a [**CHANGE REPLICATION FILTER REPLICATE\_DO\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement.

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-do-table:***channel\_1***:***db\_name.tbl\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

This option affects only statements that apply to tables. It does not affect statements that apply only to other database objects, such as stored routines. To filter statements operating on stored routines, use one or more of the --replicate-\*-db options.

[--replicate-ignore-table=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-ignore-table)***[db\_name.tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-ignore-table)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-ignore-table=name** |
| **Type** | String |

Creates a replication filter by telling the replication SQL thread not to replicate any statement that updates the specified table, even if any other tables might be updated by the same statement. To specify more than one table to ignore, use this option multiple times, once for each table. This works for cross-database updates, in contrast to [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db). See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules). You can also create such a filter by issuing a [**CHANGE REPLICATION FILTER REPLICATE\_IGNORE\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement.

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-ignore-table:***channel\_1***:***db\_name.tbl\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

This option affects only statements that apply to tables. It does not affect statements that apply only to other database objects, such as stored routines. To filter statements operating on stored routines, use one or more of the --replicate-\*-db options.

[--replicate-rewrite-db=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-rewrite-db)***[from\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-rewrite-db)***[->](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-rewrite-db)***[to\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-rewrite-db)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-rewrite-db=old\_name->new\_name** |
| **Type** | String |

Tells the replica to create a replication filter that translates the specified database to ***to\_name*** if it was ***from\_name*** on the source. Only statements involving tables are affected, not statements such as [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database), [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), and [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database).

To specify multiple rewrites, use this option multiple times. The server uses the first one with a ***from\_name*** value that matches. The database name translation is done *before* the --replicate-\* rules are tested. You can also create such a filter by issuing a [**CHANGE REPLICATION FILTER REPLICATE\_REWRITE\_DB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement.

If you use the [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db) option on the command line and the **>** character is special to your command interpreter, quote the option value. For example:

shell> **mysqld --replicate-rewrite-db="*olddb*->*newdb*"**

The effect of the [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db) option differs depending on whether statement-based or row-based binary logging format is used for the query. With statement-based format, DML statements are translated based on the current database, as specified by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement. With row-based format, DML statements are translated based on the database where the modified table exists. DDL statements are always filtered based on the current database, as specified by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement, regardless of the binary logging format.

To ensure that rewriting produces the expected results, particularly in combination with other replication filtering options, follow these recommendations when you use the [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db) option:

Create the ***from\_name*** and ***to\_name*** databases manually on the source and the replica with different names.

If you use statement-based or mixed binary logging format, do not use cross-database queries, and do not specify database names in queries. For both DDL and DML statements, rely on the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement to specify the current database, and use only the table name in queries.

If you use row-based binary logging format exclusively, for DDL statements, rely on the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement to specify the current database, and use only the table name in queries. For DML statements, you can use a fully qualified table name (***db***.***table***) if you want.

If these recommendations are followed, it is safe to use the [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db) option in combination with table-level replication filtering options such as [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table).

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. Specify the channel name followed by a colon, followed by the filter specification. The first colon is interpreted as a separator, and any subsequent colons are interpreted as literal colons. For example, to configure a channel specific replication filter on a channel named ***channel\_1***, use:

shell> **mysqld --replicate-rewrite-db=*channel\_1*:*db\_name1*->*db\_name2***

If you use a colon but do not specify a channel name, the option configures the replication filter for the default replication channel. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

[--replicate-same-server-id](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-same-server-id)

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-same-server-id[={OFF|ON}]** |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This option is for use on replicas. The default is 0 (**FALSE**). With this option set to 1 (**TRUE**), the replica does not skip events that have its own server ID. This setting is normally useful only in rare configurations.

When binary logging is enabled on a replica, the combination of the [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id) and [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) options on the replica can cause infinite loops in replication if the server is part of a circular replication topology. (In MySQL 8.0, binary logging is enabled by default, and replica update logging is the default when binary logging is enabled.) However, the use of global transaction identifiers (GTIDs) prevents this situation by skipping the execution of transactions that have already been applied. If [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set on the replica, you can start the server with this combination of options, but you cannot change to any other GTID mode while the server is running. If any other GTID mode is set, the server does not start with this combination of options.

By default, the replication I/O thread does not write binary log events to the relay log if they have the replica's server ID (this optimization helps save disk usage). If you want to use [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id), be sure to start the replica with this option before you make the replica read its own events that you want the replication SQL thread to execute.

[--replicate-wild-do-table=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-wild-do-table)***[db\_name.tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-wild-do-table)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-wild-do-table=name** |
| **Type** | String |

Creates a replication filter by telling the replication SQL thread to restrict replication to statements where any of the updated tables match the specified database and table name patterns. Patterns can contain the **%** and **\_** wildcard characters, which have the same meaning as for the [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) pattern-matching operator. To specify more than one table, use this option multiple times, once for each table. This works for cross-database updates. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules). You can also create such a filter by issuing a [**CHANGE REPLICATION FILTER REPLICATE\_WILD\_DO\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement.

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-wild-do-table:***channel\_1***:***db\_name.tbl\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

This option applies to tables, views, and triggers. It does not apply to stored procedures and functions, or events. To filter statements operating on the latter objects, use one or more of the --replicate-\*-db options.

As an example, [--replicate-wild-do-table=foo%.bar%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) replicates only updates that use a table where the database name starts with **foo** and the table name starts with **bar**.

If the table name pattern is **%**, it matches any table name and the option also applies to database-level statements ([**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database), [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), and [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database)). For example, if you use [--replicate-wild-do-table=foo%.%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table), database-level statements are replicated if the database name matches the pattern **foo%**.

To include literal wildcard characters in the database or table name patterns, escape them with a backslash. For example, to replicate all tables of a database that is named **my\_own%db**, but not replicate tables from the **my1ownAABCdb** database, you should escape the **\_** and **%** characters like this: [--replicate-wild-do-table=my\\_own\%db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table). If you use the option on the command line, you might need to double the backslashes or quote the option value, depending on your command interpreter. For example, with the **bash** shell, you would need to type [--replicate-wild-do-table=my\\\_own\\%db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table).

[--replicate-wild-ignore-table=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-wild-ignore-table)***[db\_name.tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_replicate-wild-ignore-table)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--replicate-wild-ignore-table=name** |
| **Type** | String |

Creates a replication filter which keeps the replication SQL thread from replicating a statement in which any table matches the given wildcard pattern. To specify more than one table to ignore, use this option multiple times, once for each table. This works for cross-database updates. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules). You can also create such a filter by issuing a [**CHANGE REPLICATION FILTER REPLICATE\_WILD\_IGNORE\_TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement.

This option supports channel specific replication filters, enabling multi-source replicas to use specific filters for different sources. To configure a channel specific replication filter on a channel named ***channel\_1*** use --replicate-wild-ignore:***channel\_1***:***db\_name.tbl\_name***. In this case, the first colon is interpreted as a separator and subsequent colons are literal colons. See [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters) for more information.

**Note**

Global replication filters cannot be used on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

As an example, [--replicate-wild-ignore-table=foo%.bar%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) does not replicate updates that use a table where the database name starts with **foo** and the table name starts with **bar**. For information about how matching works, see the description of the [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) option. The rules for including literal wildcard characters in the option value are the same as for [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) as well.

[--skip-slave-start](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_skip-slave-start)

|  |  |
| --- | --- |
| **Command-Line Format** | **--skip-slave-start[={OFF|ON}]** |
| **System Variable** | [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Tells the replica server not to start the replication I/O and SQL threads when the server starts. To start the threads later, use a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement.

From MySQL 8.0.24, you can use the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable in place of the command line option to allow access to this feature using MySQL Server’s privilege structure, so that database administrators do not need any privileged access to the operating system.

[--slave-skip-errors=[](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-skip-errors)***[err\_code1](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-skip-errors)***[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-skip-errors)***[err\_code2](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-skip-errors)***[,...|all|ddl\_exist\_errors]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-skip-errors)

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-skip-errors=name** |
| **System Variable** | [**slave\_skip\_errors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_skip_errors) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **OFF** |
| **Valid Values** | **OFF**  **[list of error codes]**  **all**  **ddl\_exist\_errors** |

Normally, replication stops when an error occurs on the replica, which gives you the opportunity to resolve the inconsistency in the data manually. This option causes the replication SQL thread to continue replication when a statement returns any of the errors listed in the option value.

Do not use this option unless you fully understand why you are getting errors. If there are no bugs in your replication setup and client programs, and no bugs in MySQL itself, an error that stops replication should never occur. Indiscriminate use of this option results in replicas becoming hopelessly out of synchrony with the source, with you having no idea why this has occurred.

For error codes, you should use the numbers provided by the error message in your replica's error log and in the output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). [Appendix B, *Error Messages and Common Problems*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html), lists server error codes.

The shorthand value **ddl\_exist\_errors** is equivalent to the error code list **1007,1008,1050,1051,1054,1060,1061,1068,1094,1146**.

You can also (but should not) use the very nonrecommended value of **all** to cause the replica to ignore all error messages and keeps going regardless of what happens. Needless to say, if you use **all**, there are no guarantees regarding the integrity of your data. Please do not complain (or file bug reports) in this case if the replica's data is not anywhere close to what it is on the source. *You have been warned*.

Examples:

--slave-skip-errors=1062,1053

--slave-skip-errors=all

--slave-skip-errors=ddl\_exist\_errors

[--slave-sql-verify-checksum={0|1}](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_slave-sql-verify-checksum)

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-sql-verify-checksum[={OFF|ON}]** |
| **Type** | Boolean |
| **Default Value** | **ON** |

When this option is enabled, the replica examines checksums read from the relay log. In the event of a mismatch, the replica stops with an error.

The following options are used internally by the MySQL test suite for replication testing and debugging. They are not intended for use in a production setting.

[--abort-slave-event-count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_abort-slave-event-count)

|  |  |
| --- | --- |
| **Command-Line Format** | **--abort-slave-event-count=#** |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |

When this option is set to some positive integer ***value*** other than 0 (the default) it affects replication behavior as follows: After the replication SQL thread has started, ***value*** log events are permitted to be executed; after that, the replication SQL thread does not receive any more events, just as if the network connection from the source were cut. The replication SQL thread continues to run, and the output from [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) displays **Yes** in both the **Replica\_IO\_Running** and the **Replica\_SQL\_Running** columns, but no further events are read from the relay log.

[--disconnect-slave-event-count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_disconnect-slave-event-count)

|  |  |
| --- | --- |
| **Command-Line Format** | **--disconnect-slave-event-count=#** |
| **Type** | Integer |
| **Default Value** | **0** |

**System Variables Used on Replica Servers**

The following list describes system variables for controlling replica servers. They can be set at server startup and some of them can be changed at runtime using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable). Server options used with replicas are listed earlier in this section.

**[init\_slave](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_init_slave)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--init-slave=name** |
| **System Variable** | [**init\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_init_slave) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

This variable is similar to [**init\_connect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_connect), but is a string to be executed by a replica server each time the replication SQL thread starts. The format of the string is the same as for the [**init\_connect**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_connect) variable. The setting of this variable takes effect for subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements.

**Note**

The replication SQL thread sends an acknowledgment to the client before it executes [**init\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_init_slave). Therefore, it is not guaranteed that [**init\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_init_slave) has been executed when [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) returns. See [Section 13.4.2.7, “START REPLICA | SLAVE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) for more information.

**[log\_slow\_slave\_statements](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_slow_slave_statements)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-slow-slave-statements[={OFF|ON}]** |
| **System Variable** | [**log\_slow\_slave\_statements**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slow_slave_statements) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

When the slow query log is enabled, this variable enables logging for queries that have taken more than [**long\_query\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_long_query_time) seconds to execute on the replica. Note that if row-based replication is in use ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), [**log\_slow\_slave\_statements**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slow_slave_statements) has no effect. Queries are only added to the replica's slow query log when they are logged in statement format in the binary log, that is, when [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, or when [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set and the statement is logged in statement format. Slow queries that are logged in row format when [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, or that are logged when [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, are not added to the replica's slow query log, even if [**log\_slow\_slave\_statements**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slow_slave_statements) is enabled.

Setting [**log\_slow\_slave\_statements**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slow_slave_statements) has no immediate effect. The state of the variable applies on all subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements. Also note that the global setting for [**long\_query\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_long_query_time) applies for the lifetime of the SQL thread. If you change that setting, you must stop and restart the replication SQL thread to implement the change there (for example, by issuing [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements with the **SQL\_THREAD** option).

**[master\_info\_repository](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_master_info_repository)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--master-info-repository={FILE|TABLE}** |
| **Deprecated** | 8.0.23 |
| **System Variable** | [**master\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **TABLE** |
| **Valid Values** | **FILE**  **TABLE** |

The use of this system variable is now deprecated. The setting **TABLE** is the default, and is required when multiple replication channels are configured. The alternative setting **FILE** was previously deprecated.

With the default setting, the replica records metadata about the source, consisting of status and connection information, to an **InnoDB** table in the **mysql** system database named **mysql.slave\_master\_info**. For more information on the connection metadata repository, see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs).

The **FILE** setting wrote the replica's connection metadata repository to a file, which was named master.info by default. The name could be changed using the [--master-info-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-info-file) option.

**[max\_relay\_log\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_max_relay_log_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-relay-log-size=#** |
| **System Variable** | [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1073741824** |

If a write by a replica to its relay log causes the current log file size to exceed the value of this variable, the replica rotates the relay logs (closes the current file and opens the next one). If [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) is 0, the server uses [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) for both the binary log and the relay log. If [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) is greater than 0, it constrains the size of the relay log, which enables you to have different sizes for the two logs. You must set [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) to between 4096 bytes and 1GB (inclusive), or to 0. The default value is 0. See [Section 17.2.3, “Replication Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation-details).

**[relay\_log](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log=file\_name** |
| **System Variable** | [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |

The base name for relay log files. For the default replication channel, the default base name for relay logs is ***host\_name***-relay-bin. For non-default replication channels, the default base name for relay logs is ***host\_name***-relay-bin-***channel***, where ***channel*** is the name of the replication channel recorded in this relay log.

The server writes the file in the data directory unless the base name is given with a leading absolute path name to specify a different directory. The server creates relay log files in sequence by adding a numeric suffix to the base name.

The relay log and relay log index on a replication server cannot be given the same names as the binary log and binary log index, whose names are specified by the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) and [--log-bin-index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin-index) options. The server issues an error message and does not start if the binary log and relay log file base names would be the same.

Due to the manner in which MySQL parses server options, if you specify this variable at server startup, you must supply a value; *the default base name is used only if the option is not actually specified*. If you specify the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable at server startup without specifying a value, unexpected behavior is likely to result; this behavior depends on the other options used, the order in which they are specified, and whether they are specified on the command line or in an option file. For more information about how MySQL handles server options, see [Section 4.2.2, “Specifying Program Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#program-options).

If you specify this variable, the value specified is also used as the base name for the relay log index file. You can override this behavior by specifying a different relay log index file base name using the [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variable.

When the server reads an entry from the index file, it checks whether the entry contains a relative path. If it does, the relative part of the path is replaced with the absolute path set using the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable. An absolute path remains unchanged; in such a case, the index must be edited manually to enable the new path or paths to be used.

You may find the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable useful in performing the following tasks:

Creating relay logs whose names are independent of host names.

If you need to put the relay logs in some area other than the data directory because your relay logs tend to be very large and you do not want to decrease [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size).

To increase speed by using load-balancing between disks.

You can obtain the relay log file name (and path) from the [**relay\_log\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_basename) system variable.

**[relay\_log\_basename](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_basename)**

|  |  |
| --- | --- |
| **System Variable** | [**relay\_log\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_basename) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **datadir + '/' + hostname + '-relay-bin'** |

Holds the base name and complete path to the relay log file. The maximum variable length is 256. This variable is set by the server and is read only.

**[relay\_log\_index](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_index)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-index=file\_name** |
| **System Variable** | [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **\*host\_name\*-relay-bin.index** |

The name for the relay log index file. The maximum variable length is 256. If you do not specify this variable, but the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable is specified, its value is used as the default base name for the relay log index file. If [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) is also not specified, then for the default replication channel, the default name is ***host\_name***-relay-bin.index, using the name of the host machine. For non-default replication channels, the default name is ***host\_name***-relay-bin-***channel***.index, where ***channel*** is the name of the replication channel recorded in this relay log index.

The default location for relay log files is the data directory, or any other location that was specified using the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable. You can use the [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variable to specify an alternative location, by adding a leading absolute path name to the base name to specify a different directory.

The relay log and relay log index on a replication server cannot be given the same names as the binary log and binary log index, whose names are specified by the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) and [--log-bin-index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin-index) options. The server issues an error message and does not start if the binary log and relay log file base names would be the same.

Due to the manner in which MySQL parses server options, if you specify this variable at server startup, you must supply a value; *the default base name is used only if the option is not actually specified*. If you specify the [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variable at server startup without specifying a value, unexpected behavior is likely to result; this behavior depends on the other options used, the order in which they are specified, and whether they are specified on the command line or in an option file. For more information about how MySQL handles server options, see [Section 4.2.2, “Specifying Program Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#program-options).

**[relay\_log\_info\_file](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_info_file)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-info-file=file\_name** |
| **Deprecated** | 8.0.18 |
| **System Variable** | [**relay\_log\_info\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_file) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |
| **Default Value** | **relay-log.info** |

The use of this system variable is now deprecated. It was used to set the file name for the replica's applier metadata repository if [**relay\_log\_info\_repository=FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) was set. [**relay\_log\_info\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_file) and the use of the [**relay\_log\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) system variable are deprecated because the use of a file for the applier metadata repository has been superseded by crash-safe tables. For information about the applier metadata repository, see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status).

**[relay\_log\_info\_repository](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_info_repository)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-info-repository=value** |
| **Deprecated** | 8.0.23 |
| **System Variable** | [**relay\_log\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **TABLE** |
| **Valid Values** | **FILE**  **TABLE** |

The use of this system variable is now deprecated. The setting **TABLE** is the default, and is required when multiple replication channels are configured. The **TABLE** setting for the replica's applier metadata repository is also required to make replication resilient to unexpected halts. See [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt) for more information. The alternative setting **FILE** was previously deprecated.

With the default setting, the replica stores its applier metadata repository as an **InnoDB** table in the **mysql** system database named **mysql.slave\_relay\_log\_info**. For more information on the applier metadata repository, see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs).

The **FILE** setting wrote the replica's applier metadata repository to a file, which was named relay-log.info by default. The name could be changed using the [**relay\_log\_info\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_file) system variable.

**[relay\_log\_purge](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_purge)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-purge[={OFF|ON}]** |
| **System Variable** | [**relay\_log\_purge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_purge) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Disables or enables automatic purging of relay log files as soon as they are not needed any more. The default value is 1 (**ON**).

**[relay\_log\_recovery](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_recovery)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-recovery[={OFF|ON}]** |
| **System Variable** | [**relay\_log\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

If enabled, this variable enables automatic relay log recovery immediately following server startup. The recovery process creates a new relay log file, initializes the SQL thread position to this new relay log, and initializes the I/O thread to the SQL thread position. Reading of the relay log from the source then continues.

This global variable is read-only at runtime. Its value can be set with the [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) option at replica server startup, which should be used following an unexpected halt of a replica to ensure that no possibly corrupted relay logs are processed, and must be used in order to guarantee a crash-safe replica. The default value is 0 (disabled). For information on the combination of settings on a replica that is most resilient to unexpected halts, see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

For a multithreaded replica (where [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is greater than 0), setting [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) at startup automatically handles any inconsistencies and gaps in the sequence of transactions that have been executed from the relay log. These gaps can occur when file position based replication is in use. (For more details, see [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies).) The relay log recovery process deals with gaps using the same method as the [**START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement would. When the replica reaches a consistent gap-free state, the relay log recovery process goes on to fetch further transactions from the source beginning at the SQL (applier) thread position. When GTID-based replication is in use, from MySQL 8.0.18 a multithreaded replica checks first whether **MASTER\_AUTO\_POSITION** is set to **ON**, and if it is, omits the step of calculating the transactions that should be skipped or not skipped, so that the old relay logs are not required for the recovery process.

**Note**

This variable does not affect the following Group Replication channels:

**group\_replication\_applier**

**group\_replication\_recovery**

Any other channels running on a group are affected, such as a channel which is replicating from an outside source or another group.

**[relay\_log\_space\_limit](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_relay_log_space_limit)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--relay-log-space-limit=#** |
| **System Variable** | [**relay\_log\_space\_limit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_space_limit) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The maximum amount of space to use for all relay logs.

**[replication\_optimize\_for\_static\_plugin\_config](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_replication_optimize_for_static_plugin_config)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--replication-optimize-for-static-plugin-config[={OFF|ON}]** |
| **Introduced** | 8.0.23 |
| **System Variable** | [**replication\_optimize\_for\_static\_plugin\_config**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_optimize_for_static_plugin_config) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Use shared locks, and avoid unnecessary lock acquisitions, to improve performance for semisynchronous replication. This setting and [**replication\_sender\_observe\_commit\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_sender_observe_commit_only) help as the number of replicas increases, because contention for locks can slow down performance. While this system variable is enabled, the semisynchronous replication plugin cannot be uninstalled, so you must disable the system variable before the uninstall can complete.

This system variable can be enabled before or after installing the semisynchronous replication plugin, and can be enabled while replication is running. Semisynchronous replication source servers can also get performance benefits from enabling this system variable, because they use the same locking mechanisms as the replicas.

**[replication\_sender\_observe\_commit\_only](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_replication_sender_observe_commit_only)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--replication-sender-observe-commit-only[={OFF|ON}]** |
| **Introduced** | 8.0.23 |
| **System Variable** | [**replication\_sender\_observe\_commit\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_sender_observe_commit_only) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Limit callbacks to improve performance for semisynchronous replication. This setting and [**replication\_optimize\_for\_static\_plugin\_config**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_optimize_for_static_plugin_config) help as the number of replicas increases, because contention for locks can slow down performance.

This system variable can be enabled before or after installing the semisynchronous replication plugin, and can be enabled while replication is running. Semisynchronous replication source servers can also get performance benefits from enabling this system variable, because they use the same locking mechanisms as the replicas.

**[report\_host](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_report_host)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--report-host=host\_name** |
| **System Variable** | [**report\_host**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The host name or IP address of the replica to be reported to the source during replica registration. This value appears in the output of [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) on the source server. Leave the value unset if you do not want the replica to register itself with the source.

**Note**

It is not sufficient for the source to simply read the IP address of the replica server from the TCP/IP socket after the replica connects. Due to NAT and other routing issues, that IP may not be valid for connecting to the replica from the source or other hosts.

**[report\_password](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_report_password)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--report-password=name** |
| **System Variable** | [**report\_password**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_password) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The account password of the replica to be reported to the source during replica registration. This value appears in the output of [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) on the source server if the source was started with [--show-slave-auth-info](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_show-slave-auth-info).

Although the name of this variable might imply otherwise, [**report\_password**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_password) is not connected to the MySQL user privilege system and so is not necessarily (or even likely to be) the same as the password for the MySQL replication user account.

**[report\_port](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_report_port)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--report-port=port\_num** |
| **System Variable** | [**report\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **[slave\_port]** |
| **Minimum Value** | **0** |
| **Maximum Value** | **65535** |

The TCP/IP port number for connecting to the replica, to be reported to the source during replica registration. Set this only if the replica is listening on a nondefault port or if you have a special tunnel from the source or other clients to the replica. If you are not sure, do not use this option.

The default value for this option is the port number actually used by the replica. This is also the default value displayed by [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas).

**[report\_user](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_report_user)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--report-user=name** |
| **System Variable** | [**report\_user**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_user) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |

The account user name of the replica to be reported to the source during replica registration. This value appears in the output of [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) on the source server if the source was started with [--show-slave-auth-info](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_show-slave-auth-info).

Although the name of this variable might imply otherwise, [**report\_user**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_user) is not connected to the MySQL user privilege system and so is not necessarily (or even likely to be) the same as the name of the MySQL replication user account.

**[rpl\_read\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_read_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-read-size=#** |
| **System Variable** | [**rpl\_read\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_read_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **8192** |
| **Minimum Value** | **8192** |
| **Maximum Value** | **4294967295** |

The [**rpl\_read\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_read_size) system variable controls the minimum amount of data in bytes that is read from the binary log files and relay log files. If heavy disk I/O activity for these files is impeding performance for the database, increasing the read size might reduce file reads and I/O stalls when the file data is not currently cached by the operating system.

The minimum and default value for [**rpl\_read\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_read_size) is 8192 bytes. The value must be a multiple of 4KB. Note that a buffer the size of this value is allocated for each thread that reads from the binary log and relay log files, including dump threads on sources and coordinator threads on replicas. Setting a large value might therefore have an impact on memory consumption for servers.

**[rpl\_semi\_sync\_slave\_enabled](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_slave_enabled)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-slave-enabled[={OFF|ON}]** |
| **System Variable** | [**rpl\_semi\_sync\_slave\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_enabled) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Controls whether semisynchronous replication is enabled on the replica server. To enable or disable the plugin, set this variable to **ON** or **OFF** (or 1 or 0), respectively. The default is **OFF**.

This variable is available only if the replica-side semisynchronous replication plugin is installed.

**[rpl\_semi\_sync\_slave\_trace\_level](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_semi_sync_slave_trace_level)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-semi-sync-slave-trace-level=#** |
| **System Variable** | [**rpl\_semi\_sync\_slave\_trace\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_trace_level) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **32** |

The semisynchronous replication debug trace level on the replica server. See [**rpl\_semi\_sync\_master\_trace\_level**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_trace_level) for the permissible values.

This variable is available only if the replica-side semisynchronous replication plugin is installed.

**[rpl\_stop\_slave\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_rpl_stop_slave_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--rpl-stop-slave-timeout=seconds** |
| **System Variable** | [**rpl\_stop\_slave\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_stop_slave_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **31536000** |
| **Minimum Value** | **2** |
| **Maximum Value** | **31536000** |

You can control the length of time (in seconds) that [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) waits before timing out by setting this variable. This can be used to avoid deadlocks between [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and other SQL statements using different client connections to the replica.

The maximum and default value of **rpl\_stop\_slave\_timeout** is 31536000 seconds (1 year). The minimum is 2 seconds. Changes to this variable take effect for subsequent [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statements.

This variable affects only the client that issues a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement. When the timeout is reached, the issuing client returns an error message stating that the command execution is incomplete. The client then stops waiting for the replication I/O and SQL threads to stop, but the replication threads continue to try to stop, and the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) instruction remains in effect. Once the replication threads are no longer busy, the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement is executed and the replica stops.

**[skip\_slave\_start](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_skip_slave_start)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--skip-slave-start[={OFF|ON}]** |
| **System Variable** | [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Tells the replica server not to start the replication I/O and SQL threads when the server starts. To start the threads later, use a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement.

This system variable is available from MySQL 8.0.24. It is read-only and can be set by using the **PERSIST\_ONLY** keyword or the **@@persist\_only** qualifier with the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) statement. The [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) command line option also sets this system variable. You can use the system variable in place of the command line option to allow access to this feature using MySQL Server’s privilege structure, so that database administrators do not need any privileged access to the operating system.

**[slave\_checkpoint\_group](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_checkpoint_group)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-checkpoint-group=#** |
| **System Variable** | [**slave\_checkpoint\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_group) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **512** |
| **Minimum Value** | **32** |
| **Maximum Value** | **524280** |
| **Block Size** | **8** |

Sets the maximum number of transactions that can be processed by a multithreaded replica before a checkpoint operation is called to update its status as shown by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Setting this variable has no effect on replicas for which multithreading is not enabled. Setting this variable has no immediate effect. The state of the variable applies on all subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) commands.

**Note**

Multithreaded replicas are not currently supported by NDB Cluster, which silently ignores the setting for this variable. See [Section 23.6.3, “Known Issues in NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-issues), for more information.

This variable works in combination with the [**slave\_checkpoint\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_period) system variable in such a way that, when either limit is exceeded, the checkpoint is executed and the counters tracking both the number of transactions and the time elapsed since the last checkpoint are reset.

The minimum allowed value for this variable is 32, unless the server was built using [-DWITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug), in which case the minimum value is 1. The effective value is always a multiple of 8; you can set it to a value that is not such a multiple, but the server rounds it down to the next lower multiple of 8 before storing the value. (*Exception*: No such rounding is performed by the debug server.) Regardless of how the server was built, the default value is 512, and the maximum allowed value is 524280.

**[slave\_checkpoint\_period](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_checkpoint_period)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-checkpoint-period=#** |
| **System Variable** | [**slave\_checkpoint\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_period) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **300** |
| **Minimum Value** | **1** |
| **Maximum Value** | **4294967295** |
| **Unit** | **milliseconds** |

Sets the maximum time (in milliseconds) that is allowed to pass before a checkpoint operation is called to update the status of a multithreaded replica as shown by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Setting this variable has no effect on replicas for which multithreading is not enabled. Setting this variable takes effect for all replication channels immediately, including running channels.

**Note**

Multithreaded replicas are not currently supported by NDB Cluster, which silently ignores the setting for this variable. See [Section 23.6.3, “Known Issues in NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-issues), for more information.

This variable works in combination with the [**slave\_checkpoint\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_group) system variable in such a way that, when either limit is exceeded, the checkpoint is executed and the counters tracking both the number of transactions and the time elapsed since the last checkpoint are reset.

The minimum allowed value for this variable is 1, unless the server was built using [-DWITH\_DEBUG](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#option_cmake_with_debug), in which case the minimum value is 0. Regardless of how the server was built, the default value is 300, and the maximum possible value is 4294967295 (4GB).

**[slave\_compressed\_protocol](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_compressed_protocol)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-compressed-protocol[={OFF|ON}]** |
| **Deprecated** | 8.0.18 |
| **System Variable** | [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Whether to use compression of the source/replica connection protocol if both source and replica support it. If this variable is disabled (the default), connections are uncompressed. Changes to this variable take effect on subsequent connection attempts; this includes after issuing a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement, as well as reconnections made by a running replication I/O thread.

Binary log transaction compression (available as of MySQL 8.0.20), which is activated by the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, can also be used to save bandwidth. If you use binary log transaction compression in combination with protocol compression, protocol compression has less opportunity to act on the data, but can still compress headers and those events and transaction payloads that are uncompressed. For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

As of MySQL 8.0.18, if [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) is enabled, it takes precedence over any **SOURCE\_COMPRESSION\_ALGORITHMS** | **MASTER\_COMPRESSION\_ALGORITHMS** option specified for the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement. In this case, connections to the source use **zlib** compression if both the source and replica support that algorithm. If [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) is disabled, the value of **SOURCE\_COMPRESSION\_ALGORITHMS** | **MASTER\_COMPRESSION\_ALGORITHMS** applies. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

As of MySQL 8.0.18, this system variable is deprecated. You should expect it to be removed in a future version of MySQL. See [Configuring Legacy Connection Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-legacy-configuration).

**[slave\_exec\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_exec_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-exec-mode=mode** |
| **System Variable** | [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **IDEMPOTENT** (NDB)  **STRICT** (Other) |
| **Valid Values** | **IDEMPOTENT**  **STRICT** |

Controls how a replication thread resolves conflicts and errors during replication. **IDEMPOTENT** mode causes suppression of duplicate-key and no-key-found errors; **STRICT** means no such suppression takes place.

**IDEMPOTENT** mode is intended for use in multi-source replication, circular replication, and some other special replication scenarios for NDB Cluster Replication. (See [Section 23.6.10, “NDB Cluster Replication: Bidrectional and Circular Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-multi-source), and [Section 23.6.11, “NDB Cluster Replication Conflict Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-conflict-resolution), for more information.) NDB Cluster ignores any value explicitly set for [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode), and always treats it as **IDEMPOTENT**.

In MySQL Server 8.0, **STRICT** mode is the default value.

Setting this variable takes immediate effect for all replication channels, including running channels.

For storage engines other than [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html), ***IDEMPOTENT****mode should be used only when you are absolutely sure that duplicate-key errors and key-not-found errors can safely be ignored*. It is meant to be used in fail-over scenarios for NDB Cluster where multi-source replication or circular replication is employed, and is not recommended for use in other cases.

**[slave\_load\_tmpdir](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_load_tmpdir)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-load-tmpdir=dir\_name** |
| **System Variable** | [**slave\_load\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_load_tmpdir) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Directory name |
| **Default Value** | **Value of --tmpdir** |

The name of the directory where the replica creates temporary files. Setting this variable takes effect for all replication channels immediately, including running channels. The variable value is by default equal to the value of the [**tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) system variable, or the default that applies when that system variable is not specified.

When the replication SQL thread replicates a [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement, it extracts the file to be loaded from the relay log into temporary files, and then loads these into the table. If the file loaded on the source is huge, the temporary files on the replica are huge, too. Therefore, it might be advisable to use this option to tell the replica to put temporary files in a directory located in some file system that has a lot of available space. In that case, the relay logs are huge as well, so you might also want to set the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable to place the relay logs in that file system.

The directory specified by this option should be located in a disk-based file system (not a memory-based file system) so that the temporary files used to replicate [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements can survive machine restarts. The directory also should not be one that is cleared by the operating system during the system startup process. However, replication can now continue after a restart if the temporary files have been removed.

**[slave\_max\_allowed\_packet](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_max_allowed_packet)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-max-allowed-packet=#** |
| **System Variable** | [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1073741824** |
| **Minimum Value** | **1024** |
| **Maximum Value** | **1073741824** |

This option sets the maximum packet size in bytes that the replication SQL and I/O threads can handle. Setting this variable takes effect for all replication channels immediately, including running channels. It is possible for a source to write binary log events longer than its [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) setting once the event header is added. The setting for [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) must be larger than the [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) setting on the source, so that large updates using row-based replication do not cause replication to fail.

This global variable always has a value that is a positive integer multiple of 1024; if you set it to some value that is not, the value is rounded down to the next highest multiple of 1024 for it is stored or used; setting **slave\_max\_allowed\_packet** to 0 causes 1024 to be used. (A truncation warning is issued in all such cases.) The default and maximum value is 1073741824 (1 GB); the minimum is 1024.

**[slave\_net\_timeout](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_net_timeout)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-net-timeout=#** |
| **System Variable** | [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **60** |
| **Minimum Value** | **1** |
| **Maximum Value** | **31536000** |

The number of seconds to wait for more data or a heartbeat signal from the source before the replica considers the connection broken, aborts the read, and tries to reconnect. Setting this variable has no immediate effect. The state of the variable applies on all subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) commands.

The default value is 60 seconds (one minute). The first retry occurs immediately after the timeout. The interval between retries is controlled by the **SOURCE\_CONNECT\_RETRY** | **MASTER\_CONNECT\_RETRY** option for the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, and the number of reconnection attempts is limited by the **SOURCE\_RETRY\_COUNT** | **MASTER\_RETRY\_COUNT** option.

The heartbeat interval, which stops the connection timeout occurring in the absence of data if the connection is still good, is controlled by the **SOURCE\_HEARTBEAT\_PERIOD** | **MASTER\_HEARTBEAT\_PERIOD** option for the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement. The heartbeat interval defaults to half the value of [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout), and it is recorded in the replica's connection metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table. Note that a change to the value or default setting of [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) does not automatically change the heartbeat interval, whether that has been set explicitly or is using a previously calculated default. If the connection timeout is changed, you must also issue [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) to adjust the heartbeat interval to an appropriate value so that it occurs before the connection timeout.

**[slave\_parallel\_type](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_parallel_type)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-parallel-type=value** |
| **System Variable** | [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **DATABASE** |
| **Valid Values** | **DATABASE**  **LOGICAL\_CLOCK** |

For multithreaded replicas (replicas on which [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is set to a value greater than 0), [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) specifies the policy used to decide which transactions are allowed to execute in parallel on the replica. The variable has no effect on replicas for which multithreading is not enabled. The possible values are:

**LOGICAL\_CLOCK**: Transactions that are part of the same binary log group commit on a source are applied in parallel on a replica. The dependencies between transactions are tracked based on their timestamps to provide additional parallelization where possible. When this value is set, the [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) system variable can be used on the source to specify that write sets are used for parallelization in place of timestamps, if a write set is available for the transaction and gives improved results compared to timestamps.

**DATABASE**: Transactions that update different databases are applied in parallel. This value is only appropriate if data is partitioned into multiple databases which are being updated independently and concurrently on the source. There must be no cross-database constraints, as such constraints may be violated on the replica.

When [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) is set, you can only use **LOGICAL\_CLOCK**.

When your replication topology uses multiple levels of replicas, **LOGICAL\_CLOCK** may achieve less parallelization for each level the replica is away from the source. You can reduce this effect by using [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) on the source to specify that write sets are used instead of timestamps for parallelization where possible.

When binary log transaction compression is enabled using the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, if [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **DATABASE**, all the databases affected by the transaction are mapped before the transaction is scheduled. The use of binary log transaction compression with the **DATABASE** policy can reduce parallelism compared to uncompressed transactions, which are mapped and scheduled for each event.

**[slave\_parallel\_workers](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_parallel_workers)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-parallel-workers=#** |
| **System Variable** | [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1024** |

Enables multithreading on the replica and sets the number of applier threads for executing replication transactions in parallel. When the value is a number greater than 0, the replica is a multithreaded replica with the specified number of applier threads, plus a coordinator thread to manage them. If you are using multiple replication channels, each channel has this number of threads.

**Note**

Multithreaded replicas are not currently supported by NDB Cluster, which silently ignores the setting for this variable. See [Section 23.6.3, “Known Issues in NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-issues), for more information.

Retrying of transactions is supported when multithreading is enabled on a replica. When [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order), transactions on a replica are externalized on the replica in the same order as they appear in the replica's relay log. The way in which transactions are distributed among applier threads is configured by [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type).

To disable parallel execution, set this option to 0, which gives the replica a single applier thread and no coordinator thread. With this setting, the [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) and [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) system variables have no effect and are ignored.

Setting [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) has no immediate effect. The state of the variable applies on all subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements.

**[slave\_pending\_jobs\_size\_max](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_pending_jobs_size_max)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-pending-jobs-size-max=#** |
| **System Variable** | [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value (≥ 8.0.12)** | **128M** |
| **Default Value (8.0.11)** | **16M** |
| **Minimum Value** | **1024** |
| **Maximum Value** | **16EiB** |
| **Unit** | **bytes** |
| **Block Size** | **1024** |

For multithreaded replicas, this variable sets the maximum amount of memory (in bytes) available to applier queues holding events not yet applied. Setting this variable has no effect on replicas for which multithreading is not enabled. Setting this variable has no immediate effect. The state of the variable applies on all subsequent [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) commands.

The minimum possible value for this variable is 1024 bytes; the default is 128MB. The maximum possible value is 18446744073709551615 (16 exbibytes). Values that are not exact multiples of 1024 bytes are rounded down to the next lower multiple of 1024 bytes prior to being stored.

The value of this variable is a soft limit and can be set to match the normal workload. If an unusually large event exceeds this size, the transaction is held until all the worker threads have empty queues, and then processed. All subsequent transactions are held until the large transaction has been completed.

**[slave\_preserve\_commit\_order](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_preserve_commit_order)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-preserve-commit-order[={OFF|ON}]** |
| **System Variable** | [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

For multithreaded replicas (replicas on which [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is set to a value greater than 0), setting [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) ensures that transactions are executed and committed on the replica in the same order as they appear in the replica's relay log. This prevents gaps in the sequence of transactions that have been executed from the replica's relay log, and preserves the same transaction history on the replica as on the source (with the limitations listed below). This variable has no effect on replicas for which multithreading is not enabled.

Up to and including MySQL 8.0.18, setting [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) requires that binary logging ([**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin)) and replica update logging ([**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates)) are enabled on the replica, which are the default settings from MySQL 8.0. From MySQL 8.0.19, binary logging and replica update logging are not required on the replica to set [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order), and can be disabled if wanted. In all releases, setting [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) requires that [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **LOGICAL\_CLOCK**, which is *not* the default setting. Before changing the value of [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) and [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type), the replication SQL thread (for all replication channels if you are using multiple replication channels) must be stopped.

When [**slave\_preserve\_commit\_order=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) is set, which is the default, the transactions that a multithreaded replica applies in parallel may commit out of order. Therefore, checking for the most recently executed transaction does not guarantee that all previous transactions from the source have been executed on the replica. There is a chance of gaps in the sequence of transactions that have been executed from the replica's relay log. This has implications for logging and recovery when using a multithreaded replica. See [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies) for more information.

When [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) is set, the executing worker thread waits until all previous transactions are committed before committing. While a given thread is waiting for other worker threads to commit their transactions, it reports its status as **Waiting for preceding transaction to commit**. With this mode, a multithreaded replica never enters a state that the source was not in. This supports the use of replication for read scale-out. See [Section 17.4.5, “Using Replication for Scale-Out”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-scaleout).

**Note**

[**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) does not prevent source binary log position lag, where **Exec\_master\_log\_pos** is behind the position up to which transactions have been executed. See [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies).

[**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) does not preserve the commit order and transaction history if the replica uses filters on its binary log, such as [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db).

[**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) does not preserve the order of non-transactional DML updates. These might commit before transactions that precede them in the relay log, which might result in gaps in the sequence of transactions that have been executed from the replica's relay log.

In releases before MySQL 8.0.19, [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) does not preserve the order of statements with an **IF EXISTS** clause when the object concerned does not exist. These might commit before transactions that precede them in the relay log, which might result in gaps in the sequence of transactions that have been executed from the replica's relay log.

A limitation to preserving the commit order on the replica can occur if statement-based replication is in use, and both transactional and non-transactional storage engines participate in a non-XA transaction that is rolled back on the source. Normally, non-XA transactions that are rolled back on the source are not replicated to the replica, but in this particular situation, the transaction might be replicated to the replica. If this does happen, a multithreaded replica without binary logging does not handle the transaction rollback, so the commit order on the replica diverges from the relay log order of the transactions in that case.

**[slave\_rows\_search\_algorithms](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_rows_search_algorithms)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-rows-search-algorithms=value** |
| **Deprecated** | 8.0.18 |
| **System Variable** | [**slave\_rows\_search\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_rows_search_algorithms) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Set |
| **Default Value** | **INDEX\_SCAN,HASH\_SCAN** |
| **Valid Values** | **TABLE\_SCAN,INDEX\_SCAN**  **INDEX\_SCAN,HASH\_SCAN**  **TABLE\_SCAN,HASH\_SCAN**  **TABLE\_SCAN,INDEX\_SCAN,HASH\_SCAN** (equivalent to INDEX\_SCAN,HASH\_SCAN) |

When preparing batches of rows for row-based logging and replication, this system variable controls how the rows are searched for matches, in particular whether hash scans are used. The use of this system variable is now deprecated. The default setting **INDEX\_SCAN,HASH\_SCAN** is optimal for performance and works correctly in all scenarios. See [Section 17.5.1.27, “Replication and Row Searches”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-row-searches).

**[slave\_skip\_errors](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_skip_errors)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-skip-errors=name** |
| **System Variable** | [**slave\_skip\_errors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_skip_errors) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **OFF** |
| **Valid Values** | **OFF**  **[list of error codes]**  **all**  **ddl\_exist\_errors** |

Normally, replication stops when an error occurs on the replica, which gives you the opportunity to resolve the inconsistency in the data manually. This variable causes the replication SQL thread to continue replication when a statement returns any of the errors listed in the variable value.

**[slave\_sql\_verify\_checksum](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_sql_verify_checksum)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-sql-verify-checksum[={OFF|ON}]** |
| **System Variable** | [**slave\_sql\_verify\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_sql_verify_checksum) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Cause the replication SQL thread to verify data using the checksums read from the relay log. In the event of a mismatch, the replica stops with an error. Setting this variable takes effect for all replication channels immediately, including running channels.

**Note**

The replication I/O thread always reads checksums if possible when accepting events from over the network.

**[slave\_transaction\_retries](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_transaction_retries)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-transaction-retries=#** |
| **System Variable** | [**slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10** |
| **Minimum Value** | **0** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

Sets the maximum number of times for replication SQL threads on a single-threaded or multithreaded replica to automatically retry failed transactions before stopping. Setting this variable takes effect for all replication channels immediately, including running channels. The default value is 10. Setting the variable to 0 disables automatic retrying of transactions.

If a replication SQL thread fails to execute a transaction because of an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) deadlock or because the transaction's execution time exceeded [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html)'s [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) or [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)'s [**TransactionDeadlockDetectionTimeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-transactiondeadlockdetectiontimeout) or [**TransactionInactiveTimeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-transactioninactivetimeout), it automatically retries [**slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries) times before stopping with an error. Transactions with a non-temporary error are not retried.

The Performance Schema table [**replication\_applier\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-table) shows the number of retries that took place on each replication channel, in the **COUNT\_TRANSACTIONS\_RETRIES** column. The Performance Schema table [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) shows detailed information on transaction retries by individual applier threads on a single-threaded or multithreaded replica, and identifies the errors that caused the last transaction and the transaction currently in progress to be reattempted.

**[slave\_type\_conversions](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_slave_type_conversions)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--slave-type-conversions=set** |
| **System Variable** | [**slave\_type\_conversions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_type_conversions) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Set |
| **Default Value** |  |
| **Valid Values** | **ALL\_LOSSY**  **ALL\_NON\_LOSSY**  **ALL\_SIGNED**  **ALL\_UNSIGNED** |

Controls the type conversion mode in effect on the replica when using row-based replication. Its value is a comma-delimited set of zero or more elements from the list: **ALL\_LOSSY**, **ALL\_NON\_LOSSY**, **ALL\_SIGNED**, **ALL\_UNSIGNED**. Set this variable to an empty string to disallow type conversions between the source and the replica. Setting this variable takes effect for all replication channels immediately, including running channels.

For additional information on type conversion modes applicable to attribute promotion and demotion in row-based replication, see [Row-based replication: attribute promotion and demotion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-attribute-promotion).

**[sql\_slave\_skip\_counter](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sql_slave_skip_counter)**

|  |  |
| --- | --- |
| **System Variable** | [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The number of events from the source that a replica should skip. Setting the option has no immediate effect. The variable applies to the next [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement; the next [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement also changes the value back to 0. When this variable is set to a nonzero value and there are multiple replication channels configured, the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement can only be used with the **FOR CHANNEL *channel*** clause.

This option is incompatible with GTID-based replication, and must not be set to a nonzero value when [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set. If you need to skip transactions when employing GTIDs, use [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) from the source instead. If you have enabled GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement, [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) is available. See [Section 17.1.7.3, “Skipping Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip).

**Important**

If skipping the number of events specified by setting this variable would cause the replica to begin in the middle of an event group, the replica continues to skip until it finds the beginning of the next event group and begins from that point. For more information, see [Section 17.1.7.3, “Skipping Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip).

**[sync\_master\_info](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sync_master_info)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--sync-master-info=#** |
| **System Variable** | [**sync\_master\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_master_info) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The number of events after which the replica updates the connection metadata repository. When the connection metadata repository is stored as an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table, which is the default from MySQL 8.0, it is updated after this number of events. If the connection metadata repository is stored as a file, which is deprecated from MySQL 8.0, the replica synchronizes its **master.info** file to disk (using **fdatasync()**) after this number of events. The default value is 10000, and a zero value means that the repository is never updated. Setting this variable takes effect for all replication channels immediately, including running channels.

**[sync\_relay\_log](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sync_relay_log)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--sync-relay-log=#** |
| **System Variable** | [**sync\_relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

If the value of this variable is greater than 0, the MySQL server synchronizes its relay log to disk (using **fdatasync()**) after every **sync\_relay\_log** events are written to the relay log. Setting this variable takes effect for all replication channels immediately, including running channels.

Setting **sync\_relay\_log** to 0 causes no synchronization to be done to disk; in this case, the server relies on the operating system to flush the relay log's contents from time to time as for any other file.

A value of 1 is the safest choice because in the event of an unexpected halt you lose at most one event from the relay log. However, it is also the slowest choice (unless the disk has a battery-backed cache, which makes synchronization very fast). For information on the combination of settings on a replica that is most resilient to unexpected halts, see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

**[sync\_relay\_log\_info](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sync_relay_log_info)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--sync-relay-log-info=#** |
| **System Variable** | [**sync\_relay\_log\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log_info) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **10000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

The number of transactions after which the replica updates the applier metadata repository. When the applier metadata repository is stored as an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table, which is the default from MySQL 8.0, it is updated after every transaction and this system variable is ignored. If the applier metadata repository is stored as a file, which is deprecated from MySQL 8.0, the replica synchronizes its **relay-log.info** file to disk (using **fdatasync()**) after this number of transactions. The default value for **sync\_relay\_log\_info** is 10000, and a zero value means that the file contents are only flushed by the operating system. Setting this variable takes effect for all replication channels immediately, including running channels.

**17.1.6.4 Binary Logging Options and Variables**

[Startup Options Used with Binary Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-optvars-binlog)

[System Variables Used with Binary Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sysvars-binlog)

You can use the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) options and system variables that are described in this section to affect the operation of the binary log as well as to control which statements are written to the binary log. For additional information about the binary log, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log). For additional information about using MySQL server options and system variables, see [Section 5.1.7, “Server Command Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-options), and [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

**Startup Options Used with Binary Logging**

The following list describes startup options for enabling and configuring the binary log. System variables used with binary logging are discussed later in this section.

[--binlog-row-event-max-size=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-row-event-max-size)***[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-row-event-max-size)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-row-event-max-size=#** |
| **System Variable (≥ 8.0.14)** | [**binlog\_row\_event\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_event_max_size) |
| **Scope (≥ 8.0.14)** | Global |
| **Dynamic (≥ 8.0.14)** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies (≥ 8.0.14)** | No |
| **Type** | Integer |
| **Default Value** | **8192** |
| **Minimum Value** | **256** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

When row-based binary logging is used, this setting is a soft limit on the maximum size of a row-based binary log event, in bytes. Where possible, rows stored in the binary log are grouped into events with a size not exceeding the value of this setting. If an event cannot be split, the maximum size can be exceeded. The value must be (or else gets rounded down to) a multiple of 256. The default is 8192 bytes.

[--log-bin[=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin)***[base\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin)***[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin)

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-bin=file\_name** |
| **Type** | File name |

Specifies the base name to use for binary log files. With binary logging enabled, the server logs all statements that change data to the binary log, which is used for backup and replication. The binary log is a sequence of files with a base name and numeric extension. The --log-bin option value is the base name for the log sequence. The server creates binary log files in sequence by adding a numeric suffix to the base name.

If you do not supply the --log-bin option, MySQL uses binlog as the default base name for the binary log files. For compatibility with earlier releases, if you supply the --log-bin option with no string or with an empty string, the base name defaults to ***host\_name***-bin, using the name of the host machine.

The default location for binary log files is the data directory. You can use the --log-bin option to specify an alternative location, by adding a leading absolute path name to the base name to specify a different directory. When the server reads an entry from the binary log index file, which tracks the binary log files that have been used, it checks whether the entry contains a relative path. If it does, the relative part of the path is replaced with the absolute path set using the --log-bin option. An absolute path recorded in the binary log index file remains unchanged; in such a case, the index file must be edited manually to enable a new path or paths to be used. The binary log file base name and any specified path are available as the [**log\_bin\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_basename) system variable.

In earlier MySQL versions, binary logging was disabled by default, and was enabled if you specified the --log-bin option. From MySQL 8.0, binary logging is enabled by default, whether or not you specify the --log-bin option. The exception is if you use [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) to initialize the data directory manually by invoking it with the --initialize or --initialize-insecure option, when binary logging is disabled by default. It is possible to enable binary logging in this case by specifying the --log-bin option. When binary logging is enabled, the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable, which shows the status of binary logging on the server, is set to ON.

To disable binary logging, you can specify the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) or [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option at startup. If either of these options is specified and --log-bin is also specified, the option specified later takes precedence. When binary logging is disabled, the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable is set to OFF.

When GTIDs are in use on the server, if you disable binary logging when restarting the server after an abnormal shutdown, some GTIDs are likely to be lost, causing replication to fail. In a normal shutdown, the set of GTIDs from the current binary log file is saved in the **mysql.gtid\_executed** table. Following an abnormal shutdown where this did not happen, during recovery the GTIDs are added to the table from the binary log file, provided that binary logging is still enabled. If binary logging is disabled for the server restart, the server cannot access the binary log file to recover the GTIDs, so replication cannot be started. Binary logging can be disabled safely after a normal shutdown.

The [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) and [--slave-preserve-commit-order](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) options require binary logging. If you disable binary logging, either omit these options, or specify [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) and [--skip-slave-preserve-commit-order](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order). MySQL disables these options by default when [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) or [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) is specified. If you specify [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) or [--slave-preserve-commit-order](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) together with [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) or [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin), a warning or error message is issued.

In MySQL 5.7, a server ID had to be specified when binary logging was enabled, or the server would not start. In MySQL 8.0, the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable is set to 1 by default. The server can now be started with this default server ID when binary logging is enabled, but an informational message is issued if you do not specify a server ID explicitly by setting the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable. For servers that are used in a replication topology, you must specify a unique nonzero server ID for each server.

For information on the format and management of the binary log, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

[--log-bin-index[=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin-index)***[file\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin-index)***[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_log-bin-index)

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-bin-index=file\_name** |
| **System Variable** | [**log\_bin\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_index) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |

The name for the binary log index file, which contains the names of the binary log files. By default, it has the same location and base name as the value specified for the binary log files using the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option, plus the extension .index. If you do not specify [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin), the default binary log index file name is binlog.index. If you specify [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option with no string or an empty string, the default binary log index file name is ***host\_name***-bin.index, using the name of the host machine.

For information on the format and management of the binary log, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

**Statement selection options.** The options in the following list affect which statements are written to the binary log, and thus sent by a replication source server to its replicas. There are also options for replicas that control which statements received from the source should be executed or ignored. For details, see [Section 17.1.6.3, “Replica Server Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-replica).

[--binlog-do-db=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-do-db)***[db\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-do-db)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-do-db=name** |
| **Type** | String |

This option affects binary logging in a manner similar to the way that [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) affects replication.

The effects of this option depend on whether the statement-based or row-based logging format is in use, in the same way that the effects of [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) depend on whether statement-based or row-based replication is in use. You should keep in mind that the format used to log a given statement may not necessarily be the same as that indicated by the value of [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format). For example, DDL statements such as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) are always logged as statements, without regard to the logging format in effect, so the following statement-based rules for --binlog-do-db always apply in determining whether or not the statement is logged.

**Statement-based logging.** Only those statements are written to the binary log where the default database (that is, the one selected by [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use)) is ***db\_name***. To specify more than one database, use this option multiple times, once for each database; however, doing so does *not* cause cross-database statements such as **UPDATE *some\_db.some\_table* SET foo='bar'** to be logged while a different database (or no database) is selected.

**Warning**

To specify multiple databases you *must* use multiple instances of this option. Because database names can contain commas, the list is treated as the name of a single database if you supply a comma-separated list.

An example of what does not work as you might expect when using statement-based logging: If the server is started with [--binlog-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) and you issue the following statements, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement is *not* logged:

USE prices;

UPDATE sales.january SET amount=amount+1000;

The main reason for this “just check the default database” behavior is that it is difficult from the statement alone to know whether it should be replicated (for example, if you are using multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements or multiple-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements that act across multiple databases). It is also faster to check only the default database rather than all databases if there is no need.

Another case which may not be self-evident occurs when a given database is replicated even though it was not specified when setting the option. If the server is started with --binlog-do-db=sales, the following [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement is logged even though **prices** was not included when setting --binlog-do-db:

USE sales;

UPDATE prices.discounts SET percentage = percentage + 10;

Because **sales** is the default database when the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement is issued, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) is logged.

**Row-based logging.** Logging is restricted to database ***db\_name***. Only changes to tables belonging to ***db\_name*** are logged; the default database has no effect on this. Suppose that the server is started with [--binlog-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) and row-based logging is in effect, and then the following statements are executed:

USE prices;

UPDATE sales.february SET amount=amount+100;

The changes to the **february** table in the **sales** database are logged in accordance with the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement; this occurs whether or not the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement was issued. However, when using the row-based logging format and [--binlog-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db), changes made by the following [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) are not logged:

USE prices;

UPDATE prices.march SET amount=amount-25;

Even if the **USE prices** statement were changed to **USE sales**, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement's effects would still not be written to the binary log.

Another important difference in [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) handling for statement-based logging as opposed to the row-based logging occurs with regard to statements that refer to multiple databases. Suppose that the server is started with [--binlog-do-db=db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db), and the following statements are executed:

USE db1;

UPDATE db1.table1, db2.table2 SET db1.table1.col1 = 10, db2.table2.col2 = 20;

If you are using statement-based logging, the updates to both tables are written to the binary log. However, when using the row-based format, only the changes to **table1** are logged; **table2** is in a different database, so it is not changed by the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). Now suppose that, instead of the **USE db1** statement, a **USE db4** statement had been used:

USE db4;

UPDATE db1.table1, db2.table2 SET db1.table1.col1 = 10, db2.table2.col2 = 20;

In this case, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement is not written to the binary log when using statement-based logging. However, when using row-based logging, the change to **table1** is logged, but not that to **table2**—in other words, only changes to tables in the database named by [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) are logged, and the choice of default database has no effect on this behavior.

[--binlog-ignore-db=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-ignore-db)***[db\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-ignore-db)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-ignore-db=name** |
| **Type** | String |

This option affects binary logging in a manner similar to the way that [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) affects replication.

The effects of this option depend on whether the statement-based or row-based logging format is in use, in the same way that the effects of [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) depend on whether statement-based or row-based replication is in use. You should keep in mind that the format used to log a given statement may not necessarily be the same as that indicated by the value of [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format). For example, DDL statements such as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) are always logged as statements, without regard to the logging format in effect, so the following statement-based rules for --binlog-ignore-db always apply in determining whether or not the statement is logged.

**Statement-based logging.** Tells the server to not log any statement where the default database (that is, the one selected by [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use)) is ***db\_name***.

When there is no default database, no --binlog-ignore-db options are applied, and such statements are always logged. (Bug #11829838, Bug #60188)

**Row-based format.** Tells the server not to log updates to any tables in the database ***db\_name***. The current database has no effect.

When using statement-based logging, the following example does not work as you might expect. Suppose that the server is started with [--binlog-ignore-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) and you issue the following statements:

USE prices;

UPDATE sales.january SET amount=amount+1000;

The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement *is* logged in such a case because [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) applies only to the default database (determined by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement). Because the **sales** database was specified explicitly in the statement, the statement has not been filtered. However, when using row-based logging, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement's effects are *not* written to the binary log, which means that no changes to the **sales.january** table are logged; in this instance, [--binlog-ignore-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) causes *all* changes made to tables in the source's copy of the **sales** database to be ignored for purposes of binary logging.

To specify more than one database to ignore, use this option multiple times, once for each database. Because database names can contain commas, the list is treated as the name of a single database if you supply a comma-separated list.

You should not use this option if you are using cross-database updates and you do not want these updates to be logged.

**Checksum options.** MySQL supports reading and writing of binary log checksums. These are enabled using the two options listed here:

[--binlog-checksum={NONE|CRC32}](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_binlog-checksum)

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-checksum=type** |
| **Type** | String |
| **Default Value** | **CRC32** |
| **Valid Values** | **NONE**  **CRC32** |

Enabling this option causes the source to write checksums for events written to the binary log. Set to **NONE** to disable, or the name of the algorithm to be used for generating checksums; currently, only CRC32 checksums are supported, and CRC32 is the default. You cannot change the setting for this option within a transaction.

To control reading of checksums by the replica (from the relay log), use the [--slave-sql-verify-checksum](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_slave-sql-verify-checksum) option.

**Testing and debugging options.** The following binary log options are used in replication testing and debugging. They are not intended for use in normal operations.

[--max-binlog-dump-events=](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_max-binlog-dump-events)***[N](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_max-binlog-dump-events)***

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-binlog-dump-events=#** |
| **Type** | Integer |
| **Default Value** | **0** |

This option is used internally by the MySQL test suite for replication testing and debugging.

[--sporadic-binlog-dump-fail](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "option_mysqld_sporadic-binlog-dump-fail)

|  |  |
| --- | --- |
| **Command-Line Format** | **--sporadic-binlog-dump-fail[={OFF|ON}]** |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This option is used internally by the MySQL test suite for replication testing and debugging.

**System Variables Used with Binary Logging**

The following list describes system variables for controlling binary logging. They can be set at server startup and some of them can be changed at runtime using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable). Server options used to control binary logging are listed earlier in this section.

**[binlog\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-cache-size=#** |
| **System Variable** | [**binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_cache_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **32768** |
| **Minimum Value** | **4096** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

The size of the memory buffer to hold changes to the binary log during a transaction. The value must be a multiple of 4096.

When binary logging is enabled on the server (with the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable set to ON), a binary log cache is allocated for each client if the server supports any transactional storage engines. If the data for the transaction exceeds the space in the memory buffer, the excess data is stored in a temporary file. When binary log encryption is active on the server, the memory buffer is not encrypted, but (from MySQL 8.0.17) any temporary file used to hold the binary log cache is encrypted. After each transaction is committed, the binary log cache is reset by clearing the memory buffer and truncating the temporary file if used.

If you often use large transactions, you can increase this cache size to get better performance by reducing or eliminating the need to write to temporary files. The [**Binlog\_cache\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_cache_use) and [**Binlog\_cache\_disk\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_cache_disk_use) status variables can be useful for tuning the size of this variable. See [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

**binlog\_cache\_size** sets the size for the transaction cache only; the size of the statement cache is governed by the [**binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_stmt_cache_size) system variable.

**[binlog\_checksum](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_checksum)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-checksum=name** |
| **System Variable** | [**binlog\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Default Value** | **CRC32** |
| **Valid Values** | **NONE**  **CRC32** |

When enabled, this variable causes the source to write a checksum for each event in the binary log. **binlog\_checksum** supports the values **NONE** (which disables checksums) and **CRC32**. The default is **CRC32**. When **binlog\_checksum** is disabled (value **NONE**), the server verifies that it is writing only complete events to the binary log by writing and checking the event length (rather than a checksum) for each event.

Setting this variable on the source to a value unrecognized by the replica causes the replica to set its own **binlog\_checksum** value to **NONE**, and to stop replication with an error. If backward compatibility with older replicas is a concern, you may want to set the value explicitly to **NONE**.

Up to and including MySQL 8.0.20, Group Replication cannot make use of checksums and does not support their presence in the binary log, so you must set [**binlog\_checksum=NONE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum) when configuring a server instance to become a group member. From MySQL 8.0.21, Group Replication supports checksums, so group members may use the default setting.

Changing the value of **binlog\_checksum** causes the binary log to be rotated, because checksums must be written for an entire binary log file, and never for only part of one. You cannot change the value of **binlog\_checksum** within a transaction.

When binary log transaction compression is enabled using the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, checksums are not written for individual events in a compressed transaction payload. Instead a checksum is written for the GTID event, and a checksum for the compressed **Transaction\_payload\_event**.

**[binlog\_direct\_non\_transactional\_updates](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_direct_non_transactional_updates)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-direct-non-transactional-updates[={OFF|ON}]** |
| **System Variable** | [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Due to concurrency issues, a replica can become inconsistent when a transaction contains updates to both transactional and nontransactional tables. MySQL tries to preserve causality among these statements by writing nontransactional statements to the transaction cache, which is flushed upon commit. However, problems arise when modifications done to nontransactional tables on behalf of a transaction become immediately visible to other connections because these changes may not be written immediately into the binary log.

The [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) variable offers one possible workaround to this issue. By default, this variable is disabled. Enabling [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) causes updates to nontransactional tables to be written directly to the binary log, rather than to the transaction cache.

As of MySQL 8.0.14, setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

[***binlog\_direct\_non\_transactional\_updates***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates)*works only for statements that are replicated using the statement-based binary logging format*; that is, it works only when the value of [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is **STATEMENT**, or when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is **MIXED** and a given statement is being replicated using the statement-based format. This variable has no effect when the binary log format is **ROW**, or when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **MIXED** and a given statement is replicated using the row-based format.

**Important**

Before enabling this variable, you must make certain that there are no dependencies between transactional and nontransactional tables; an example of such a dependency would be the statement **INSERT INTO myisam\_table SELECT \* FROM innodb\_table**. Otherwise, such statements are likely to cause the replica to diverge from the source.

This variable has no effect when the binary log format is **ROW** or **MIXED**.

**[binlog\_encryption](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_encryption)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-encryption[={OFF|ON}]** |
| **Introduced** | 8.0.14 |
| **System Variable** | [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables encryption for binary log files and relay log files on this server. **OFF** is the default. **ON** sets encryption on for binary log files and relay log files. Binary logging does not need to be enabled on the server to enable encryption, so you can encrypt the relay log files on a replica that has no binary log. To use encryption, a keyring plugin must be installed and configured to supply MySQL Server's keyring service. For instructions to do this, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring). Any supported keyring plugin can be used to store binary log encryption keys.

When you first start the server with binary log encryption enabled, a new binary log encryption key is generated before the binary log and relay logs are initialized. This key is used to encrypt a file password for each binary log file (if the server has binary logging enabled) and relay log file (if the server has replication channels), and further keys generated from the file passwords are used to encrypt the data in the files. Relay log files are encrypted for all channels, including Group Replication applier channels and new channels that are created after encryption is activated. The binary log index file and relay log index file are never encrypted.

If you activate encryption while the server is running, a new binary log encryption key is generated at that time. The exception is if encryption was active previously on the server and was then disabled, in which case the binary log encryption key that was in use before is used again. The binary log file and relay log files are rotated immediately, and file passwords for the new files and all subsequent binary log files and relay log files are encrypted using this binary log encryption key. Existing binary log files and relay log files still present on the server are not automatically encrypted, but you can purge them if they are no longer needed.

If you deactivate encryption by changing the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable to **OFF**, the binary log file and relay log files are rotated immediately and all subsequent logging is unencrypted. Previously encrypted files are not automatically decrypted, but the server is still able to read them. The **BINLOG\_ENCRYPTION\_ADMIN** privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege) is required to activate or deactivate encryption while the server is running. Group Replication applier channels are not included in the relay log rotation request, so unencrypted logging for these channels does not start until their logs are rotated in normal use.

For more information on binary log file and relay log file encryption, see [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption).

**[binlog\_error\_action](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_error_action)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-error-action[=value]** |
| **System Variable** | [**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **ABORT\_SERVER** |
| **Valid Values** | **IGNORE\_ERROR**  **ABORT\_SERVER** |

Controls what happens when the server encounters an error such as not being able to write to, flush or synchronize the binary log, which can cause the source's binary log to become inconsistent and replicas to lose synchronization.

This variable defaults to **ABORT\_SERVER**, which makes the server halt logging and shut down whenever it encounters such an error with the binary log. On restart, recovery proceeds as in the case of an unexpected server halt (see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt)).

When **binlog\_error\_action** is set to **IGNORE\_ERROR**, if the server encounters such an error it continues the ongoing transaction, logs the error then halts logging, and continues performing updates. To resume binary logging [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) must be enabled again, which requires a server restart. This setting provides backward compatibility with older versions of MySQL.

**[binlog\_expire\_logs\_seconds](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_expire_logs_seconds)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-expire-logs-seconds=#** |
| **System Variable** | [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **2592000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

Sets the binary log expiration period in seconds. After their expiration period ends, binary log files can be automatically removed. Possible removals happen at startup and when the binary log is flushed. Log flushing occurs as indicated in [Section 5.4, “MySQL Server Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-logs).

The default binary log expiration period is 2592000 seconds, which equals 30 days (30\*24\*60\*60 seconds). The default applies if neither [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) nor the deprecated system variable [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) has a value set at startup. If a non-zero value for one of the variables [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) or [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) is set at startup, this value is used as the binary log expiration period. If a non-zero value for both of those variables is set at startup, the value for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is used as the binary log expiration period, and the value for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) is ignored with a warning message.

To disable automatic purging of the binary log, specify a value of 0 explicitly for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds), and do not specify a value for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days). For compatibility with earlier releases, automatic purging is also disabled if you specify a value of 0 explicitly for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) and do not specify a value for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds). In that case, the default for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is not applied.

To remove binary log files manually, use the [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) statement. See [Section 13.4.1.1, “PURGE BINARY LOGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs).

**[binlog\_format](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_format)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-format=format** |
| **System Variable** | [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **ROW** |
| **Valid Values** | **ROW**  **STATEMENT**  **MIXED** |

This system variable sets the binary logging format, and can be any one of **STATEMENT**, **ROW**, or **MIXED**. See [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats). The setting takes effect when binary logging is enabled on the server, which is the case when the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable is set to **ON**. From MySQL 8.0, binary logging is enabled by default.

[**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) can be set at startup or at runtime, except that under some conditions, changing this variable at runtime is not possible or causes replication to fail, as described later.

The default is **ROW**. *Exception*: In NDB Cluster, the default is **MIXED**; statement-based replication is not supported for NDB Cluster.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

The rules governing when changes to this variable take effect and how long the effect lasts are the same as for other MySQL server system variables. For more information, see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

When **MIXED** is specified, statement-based replication is used, except for cases where only row-based replication is guaranteed to lead to proper results. For example, this happens when statements contain user-defined functions (UDF) or the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function.

For details of how stored programs (stored procedures and functions, triggers, and events) are handled when each binary logging format is set, see [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

There are exceptions when you cannot switch the replication format at runtime:

The replication format cannot be changed from within a stored function or a trigger.

If a session has open temporary tables, the replication format cannot be changed for the session (**SET @@SESSION.binlog\_format**).

If any replication channel has open temporary tables, the replication format cannot be changed globally (**SET @@GLOBAL.binlog\_format** or **SET @@PERSIST.binlog\_format**).

If any replication channel applier thread is currently running, the replication format cannot be changed globally (**SET @@GLOBAL.binlog\_format** or **SET @@PERSIST.binlog\_format**).

Trying to switch the replication format in any of these cases (or attempting to set the current replication format) results in an error. You can, however, use **PERSIST\_ONLY** (**SET @@PERSIST\_ONLY.binlog\_format**) to change the replication format at any time, because this action does not modify the runtime global system variable value, and takes effect only after a server restart.

Switching the replication format at runtime is not recommended when any temporary tables exist, because temporary tables are logged only when using statement-based replication, whereas with row-based replication and mixed replication, they are not logged.

Changing the logging format on a replication source server does not cause a replica to change its logging format to match. Switching the replication format while replication is ongoing can cause issues if a replica has binary logging enabled, and the change results in the replica using **STATEMENT** format logging while the source is using **ROW** or **MIXED** format logging. A replica is not able to convert binary log entries received in **ROW** logging format to **STATEMENT** format for use in its own binary log, so this situation can cause replication to fail. For more information, see [Section 5.4.4.2, “Setting The Binary Log Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-setting).

The binary log format affects the behavior of the following server options:

[--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db)

[--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db)

[--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db)

[--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db)

These effects are discussed in detail in the descriptions of the individual options.

**[binlog\_group\_commit\_sync\_delay](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_group_commit_sync_delay)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-group-commit-sync-delay=#** |
| **System Variable** | [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000000** |

Controls how many microseconds the binary log commit waits before synchronizing the binary log file to disk. By default [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) is set to 0, meaning that there is no delay. Setting [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) to a microsecond delay enables more transactions to be synchronized together to disk at once, reducing the overall time to commit a group of transactions because the larger groups require fewer time units per group.

When [**sync\_binlog=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) or [**sync\_binlog=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) is set, the delay specified by [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) is applied for every binary log commit group before synchronization (or in the case of [**sync\_binlog=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog), before proceeding). When [**sync\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) is set to a value *n* greater than 1, the delay is applied after every *n* binary log commit groups.

Setting [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) can increase the number of parallel committing transactions on any server that has (or might have after a failover) a replica, and therefore can increase parallel execution on the replicas. To benefit from this effect, the replica servers must have [**slave\_parallel\_type=LOGICAL\_CLOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) set, and the effect is more significant when [**binlog\_transaction\_dependency\_tracking=COMMIT\_ORDER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) is also set. It is important to take into account both the source's throughput and the replicas' throughput when you are tuning the setting for [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay).

Setting [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) can also reduce the number of **fsync()** calls to the binary log on any server (source or replica) that has a binary log.

Note that setting [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) increases the latency of transactions on the server, which might affect client applications. Also, on highly concurrent workloads, it is possible for the delay to increase contention and therefore reduce throughput. Typically, the benefits of setting a delay outweigh the drawbacks, but tuning should always be carried out to determine the optimal setting.

**[binlog\_group\_commit\_sync\_no\_delay\_count](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_group_commit_sync_no_delay_count)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-group-commit-sync-no-delay-count=#** |
| **System Variable** | [**binlog\_group\_commit\_sync\_no\_delay\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_no_delay_count) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **1000000** |

The maximum number of transactions to wait for before aborting the current delay as specified by [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay). If [**binlog\_group\_commit\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_group_commit_sync_delay) is set to 0, then this option has no effect.

**[binlog\_max\_flush\_queue\_time](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_max_flush_queue_time)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-max-flush-queue-time=#** |
| **Deprecated** | Yes |
| **System Variable** | [**binlog\_max\_flush\_queue\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_max_flush_queue_time) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **100000** |

**binlog\_max\_flush\_queue\_time** is deprecated, and is marked for eventual removal in a future MySQL release. Formerly, this system variable controlled the time in microseconds to continue reading transactions from the flush queue before proceeding with group commit. It no longer has any effect.

**[binlog\_order\_commits](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_order_commits)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-order-commits[={OFF|ON}]** |
| **System Variable** | [**binlog\_order\_commits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_order_commits) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

When this variable is enabled on a replication source server (which is the default), transaction commit instructions issued to storage engines are serialized on a single thread, so that transactions are always committed in the same order as they are written to the binary log. Disabling this variable permits transaction commit instructions to be issued using multiple threads. Used in combination with binary log group commit, this prevents the commit rate of a single transaction being a bottleneck to throughput, and might therefore produce a performance improvement.

Transactions are written to the binary log at the point when all the storage engines involved have confirmed that the transaction is prepared to commit. The binary log group commit logic then commits a group of transactions after their binary log write has taken place. When [**binlog\_order\_commits**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_order_commits) is disabled, because multiple threads are used for this process, transactions in a commit group might be committed in a different order from their order in the binary log. (Transactions from a single client always commit in chronological order.) In many cases this does not matter, as operations carried out in separate transactions should produce consistent results, and if that is not the case, a single transaction ought to be used instead.

If you want to ensure that the transaction history on the source and on a multithreaded replica remains identical, set [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) on the replica.

**[binlog\_rotate\_encryption\_master\_key\_at\_startup](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_rotate_encryption_master_key_at_startup)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-rotate-encryption-master-key-at-startup[={OFF|ON}]** |
| **Introduced** | 8.0.14 |
| **System Variable** | [**binlog\_rotate\_encryption\_master\_key\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rotate_encryption_master_key_at_startup) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Specifies whether or not the binary log master key is rotated at server startup. The binary log master key is the binary log encryption key that is used to encrypt file passwords for the binary log files and relay log files on the server. When a server is started for the first time with binary log encryption enabled ([**binlog\_encryption=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption)), a new binary log encryption key is generated and used as the binary log master key. If the [**binlog\_rotate\_encryption\_master\_key\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rotate_encryption_master_key_at_startup) system variable is also set to **ON**, whenever the server is restarted, a further binary log encryption key is generated and used as the binary log master key for all subsequent binary log files and relay log files. If the [**binlog\_rotate\_encryption\_master\_key\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rotate_encryption_master_key_at_startup) system variable is set to **OFF**, which is the default, the existing binary log master key is used again after the server restarts. For more information on binary log encryption keys and the binary log master key, see [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption).

**[binlog\_row\_event\_max\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_row_event_max_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-row-event-max-size=#** |
| **System Variable (≥ 8.0.14)** | [**binlog\_row\_event\_max\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_event_max_size) |
| **Scope (≥ 8.0.14)** | Global |
| **Dynamic (≥ 8.0.14)** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies (≥ 8.0.14)** | No |
| **Type** | Integer |
| **Default Value** | **8192** |
| **Minimum Value** | **256** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

When row-based binary logging is used, this setting is a soft limit on the maximum size of a row-based binary log event, in bytes. Where possible, rows stored in the binary log are grouped into events with a size not exceeding the value of this setting. If an event cannot be split, the maximum size can be exceeded. The value must be (or else gets rounded down to) a multiple of 256. The default is 8192 bytes.

This global system variable is read-only and can be set only at server startup. Its value can therefore only be modified by using the **PERSIST\_ONLY** keyword or the **@@persist\_only** qualifier with the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) statement.

**[binlog\_row\_image](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_row_image)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-row-image=image\_type** |
| **System Variable** | [**binlog\_row\_image**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_image) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **full** |
| **Valid Values** | **full** (Log all columns)  **minimal** (Log only changed columns, and columns needed to identify rows)  **noblob** (Log all columns, except for unneeded BLOB and TEXT columns) |

For MySQL row-based replication, this variable determines how row images are written to the binary log.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

In MySQL row-based replication, each row change event contains two images, a “before” image whose columns are matched against when searching for the row to be updated, and an “after” image containing the changes. Normally, MySQL logs full rows (that is, all columns) for both the before and after images. However, it is not strictly necessary to include every column in both images, and we can often save disk, memory, and network usage by logging only those columns which are actually required.

**Note**

When deleting a row, only the before image is logged, since there are no changed values to propagate following the deletion. When inserting a row, only the after image is logged, since there is no existing row to be matched. Only when updating a row are both the before and after images required, and both written to the binary log.

For the before image, it is necessary only that the minimum set of columns required to uniquely identify rows is logged. If the table containing the row has a primary key, then only the primary key column or columns are written to the binary log. Otherwise, if the table has a unique key all of whose columns are **NOT NULL**, then only the columns in the unique key need be logged. (If the table has neither a primary key nor a unique key without any **NULL** columns, then all columns must be used in the before image, and logged.) In the after image, it is necessary to log only the columns which have actually changed.

You can cause the server to log full or minimal rows using the **binlog\_row\_image** system variable. This variable actually takes one of three possible values, as shown in the following list:

**full**: Log all columns in both the before image and the after image.

**minimal**: Log only those columns in the before image that are required to identify the row to be changed; log only those columns in the after image where a value was specified by the SQL statement, or generated by auto-increment.

**noblob**: Log all columns (same as **full**), except for [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns that are not required to identify rows, or that have not changed.

**Note**

This variable is not supported by NDB Cluster; setting it has no effect on the logging of [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables.

The default value is **full**.

When using **minimal** or **noblob**, deletes and updates are guaranteed to work correctly for a given table if and only if the following conditions are true for both the source and destination tables:

All columns must be present and in the same order; each column must use the same data type as its counterpart in the other table.

The tables must have identical primary key definitions.

(In other words, the tables must be identical with the possible exception of indexes that are not part of the tables' primary keys.)

If these conditions are not met, it is possible that the primary key column values in the destination table may prove insufficient to provide a unique match for a delete or update. In this event, no warning or error is issued; the source and replica silently diverge, thus breaking consistency.

Setting this variable has no effect when the binary logging format is **STATEMENT**. When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is **MIXED**, the setting for **binlog\_row\_image** is applied to changes that are logged using row-based format, but this setting has no effect on changes logged as statements.

Setting **binlog\_row\_image** on either the global or session level does not cause an implicit commit; this means that this variable can be changed while a transaction is in progress without affecting the transaction.

**[binlog\_row\_metadata](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_row_metadata)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-row-metadata=metadata\_type** |
| **System Variable** | [**binlog\_row\_metadata**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_metadata) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **MINIMAL** |
| **Valid Values** | **FULL** (All metadata is included)  **MINIMAL** (Limit included metadata) |

Configures the amount of table metadata added to the binary log when using row-based logging. When set to **MINIMAL**, the default, only metadata related to **SIGNED** flags, column character set and geometry types are logged. When set to **FULL** complete metadata for tables is logged, such as column name, [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or **SET** string values, **PRIMARY KEY** information, and so on.

The extended metadata serves the following purposes:

Replicas use the metadata to transfer data when its table structure is different from the source's.

External software can use the metadata to decode row events and store the data into external databases, such as a data warehouse.

**[binlog\_row\_value\_options](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_row_value_options)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-row-value-options=#** |
| **System Variable** | [**binlog\_row\_value\_options**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_value_options) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Set |
| **Default Value** | **''** |
| **Valid Values** | **PARTIAL\_JSON** |

When set to **PARTIAL\_JSON**, this enables use of a space-efficient binary log format for updates that modify only a small portion of a JSON document, which causes row-based replication to write only the modified parts of the JSON document to the after-image for the update in the binary log (rather than writing the full document). This works for an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement which modifies a JSON column using any sequence of [**JSON\_SET()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-set), [**JSON\_REPLACE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-replace), and [**JSON\_REMOVE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-remove). If the modification requires more space than the full document, or if the server is unable to generate a partial update, the full document is used instead.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

**PARTIAL\_JSON** is the only supported value; to unset **binlog\_row\_value\_options**, set its value to the empty string.

**binlog\_row\_value\_options=PARTIAL\_JSON** takes effect only when binary logging is enabled and [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **ROW** or **MIXED**. Statement-based replication *always* logs only the modified parts of the JSON document, regardless of any value set for **binlog\_row\_value\_options**. To maximize the amount of space saved, use [**binlog\_row\_image=NOBLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_image) or **binlog\_row\_image=MINIMAL** together with this option. **binlog\_row\_image=FULL** saves less space than either of these, since the full JSON document is stored in the before-image, and the partial update is stored only in the after-image.

[**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output includes partial JSON updates in the form of events encoded as base-64 strings using [**BINLOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog) statements. If the [--verbose](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_verbose) option is specified, [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) displays the partial JSON updates as readable JSON using pseudo-SQL statements.

MySQL Replication generates an error if a modification cannot be applied to the JSON document on the replica. This includes a failure to find the path. Be aware that, even with this and other safety checks, if a JSON document on a replica has diverged from that on the source and a partial update is applied, it remains theoretically possible to produce a valid but unexpected JSON document on the replica.

**binlog\_rows\_query\_log\_events**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-rows-query-log-events[={OFF|ON}]** |
| **System Variable** | [**binlog\_rows\_query\_log\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rows_query_log_events) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This system variable affects row-based logging only. When enabled, it causes the server to write informational log events such as row query log events into its binary log. This information can be used for debugging and related purposes, such as obtaining the original query issued on the source when it cannot be reconstructed from the row updates.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

These informational events are normally ignored by MySQL programs reading the binary log and so cause no issues when replicating or restoring from backup. To view them, increase the verbosity level by using mysqlbinlog's [--verbose](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_verbose) option twice, either as -vv or --verbose --verbose.

**[binlog\_stmt\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_stmt_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-stmt-cache-size=#** |
| **System Variable** | [**binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_stmt_cache_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **32768** |
| **Minimum Value** | **4096** |
| **Maximum Value (64-bit platforms)** | **18446744073709551615** |
| **Maximum Value (32-bit platforms)** | **4294967295** |

The size of the memory buffer for the binary log to hold nontransactional statements issued during a transaction. The value must be a multiple of 4096.

When binary logging is enabled on the server (with the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable set to ON), separate binary log transaction and statement caches are allocated for each client if the server supports any transactional storage engines. If the data for the nontransactional statements used in the transaction exceeds the space in the memory buffer, the excess data is stored in a temporary file. When binary log encryption is active on the server, the memory buffer is not encrypted, but (from MySQL 8.0.17) any temporary file used to hold the binary log cache is encrypted. After each transaction is committed, the binary log statement cache is reset by clearing the memory buffer and truncating the temporary file if used.

If you often use large nontransactional statements during transactions, you can increase this cache size to get better performance by reducing or eliminating the need to write to temporary files. The [**Binlog\_stmt\_cache\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_stmt_cache_use) and [**Binlog\_stmt\_cache\_disk\_use**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Binlog_stmt_cache_disk_use) status variables can be useful for tuning the size of this variable. See [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

The [**binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_cache_size) system variable sets the size for the transaction cache.

**[binlog\_transaction\_compression](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_transaction_compression)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-transaction-compression[={OFF|ON}]** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enables compression for transactions that are written to binary log files on this server. **OFF** is the default. Use the [**binlog\_transaction\_compression\_level\_zstd**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression_level_zstd) system variable to set the level for the zstd algorithm that is used for compression.

When binary log transaction compression is enabled, transaction payloads are compressed and then written to the binary log file as a single event (**Transaction\_payload\_event**). Compressed transaction payloads remain in a compressed state while they are sent in the replication stream to replicas, other Group Replication group members, or clients such as [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), and are written to the relay log still in their compressed state. Binary log transaction compression therefore saves storage space both on the originator of the transaction and on the recipient (and for their backups), and saves network bandwidth when the transactions are sent between server instances.

For **binlog\_transaction\_compression=ON** to have a direct effect, binary logging must be enabled on the server. When a MySQL server instance has no binary log, if it is at a release from MySQL 8.0.20, it can receive, handle, and display compressed transaction payloads regardless of its value for [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression). Compressed transaction payloads received by such server instances are written in their compressed state to the relay log, so they benefit indirectly from compression carried out by other servers in the replication topology.

This system variable cannot be changed within the context of a transaction. Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

For more information on binary log transaction compression, including details of what events are and are not compressed, and changes in behavior when transaction compression is in use, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

**[binlog\_transaction\_compression\_level\_zstd](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_transaction_compression_level_zstd)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-transaction-compression-level-zstd=#** |
| **Introduced** | 8.0.20 |
| **System Variable** | [**binlog\_transaction\_compression\_level\_zstd**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression_level_zstd) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **3** |
| **Minimum Value** | **1** |
| **Maximum Value** | **22** |

Sets the compression level for binary log transaction compression on this server, which is enabled by the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable. The value is an integer that determines the compression effort, from 1 (the lowest effort) to 22 (the highest effort). If you do not specify this system variable, the compression level is set to 3.

As the compression level increases, the data compression ratio increases, which reduces the storage space and network bandwidth required for the transaction payload. However, the effort required for data compression also increases, taking time and CPU and memory resources on the originating server. Increases in the compression effort do not have a linear relationship to increases in the data compression ratio.

This system variable cannot be changed within the context of a transaction. Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

**[binlog\_transaction\_dependency\_tracking](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_transaction_dependency_tracking)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-transaction-dependency-tracking=value** |
| **System Variable** | [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **COMMIT\_ORDER** |
| **Valid Values** | **COMMIT\_ORDER**  **WRITESET**  **WRITESET\_SESSION** |

For a replication source server that has multithreaded replicas (replicas on which [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is set to a value greater than 0), [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) specifies the source of dependency information that the source records in the binary log to help replicas determine which transactions can be executed in parallel. The possible values are:

**COMMIT\_ORDER**: Dependency information is generated from the source's commit timestamps. This is the default.

**WRITESET**: Dependency information is generated from the source's write set, and any transactions that write different tuples can be parallelized.

**WRITESET\_SESSION**: Dependency information is generated from the source's write set, and any transactions that write different tuples can be parallelized, with the exception that no two updates from the same session can be reordered.

In **WRITESET** or **WRITESET\_SESSION** mode, transactions can commit out of order unless you also set [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order).

For some transactions, the **WRITESET** and **WRITESET\_SESSION** modes cannot improve on the results that would have been returned in **COMMIT\_ORDER** mode. This is the case for transactions that have empty or partial write sets, transactions that update tables without primary or unique keys, and transactions that update parent tables in a foreign key relationship. In these situations, the source uses **COMMIT\_ORDER** mode to generate the dependency information instead.

To set **WRITESET** or **WRITESET\_SESSION** as the value for [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking), [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) must be set to specify an algorithm (not set to **OFF**). The default in MySQL 8.0 is that [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) is set to **XXHASH64**. The value that you select for [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) cannot be changed again while the value of [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) remains as **WRITESET** or **WRITESET\_SESSION**.

The number of row hashes to be kept and checked for the latest transaction to have changed a given row is determined by the value of [**binlog\_transaction\_dependency\_history\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_history_size).

Group Replication carries out its own parallelization after certification when applying transactions from the relay log, independently of the value set for [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking). However, the value of [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) does affect how transactions are written to the binary logs on Group Replication members. The dependency information in those logs is used to assist the process of state transfer from a donor's binary log for distributed recovery, which takes place whenever a member joins or rejoins the group. For that process, setting [**binlog\_transaction\_dependency\_tracking=WRITESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) can improve performance for a group member, depending on the group's workload.

**[binlog\_transaction\_dependency\_history\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_transaction_dependency_history_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-transaction-dependency-history-size=#** |
| **System Variable** | [**binlog\_transaction\_dependency\_history\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_history_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **25000** |
| **Minimum Value** | **1** |
| **Maximum Value** | **1000000** |

Sets an upper limit on the number of row hashes which are kept in memory and used for looking up the transaction that last modified a given row. Once this number of hashes has been reached, the history is purged.

**[expire\_logs\_days](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_expire_logs_days)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--expire-logs-days=#** |
| **Deprecated** | Yes |
| **System Variable** | [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **0** |
| **Minimum Value** | **0** |
| **Maximum Value** | **99** |

Specifies the number of days before automatic removal of binary log files. [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) is deprecated, and you should expect it to be removed in a future release. Instead, use [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds), which sets the binary log expiration period in seconds. If you do not set a value for either system variable, the default expiration period is 30 days. Possible removals happen at startup and when the binary log is flushed. Log flushing occurs as indicated in [Section 5.4, “MySQL Server Logs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-logs).

Any non-zero value that you specify for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) is ignored if [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is also specified, and the value of [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is used instead as the binary log expiration period. A warning message is issued in this situation. A non-zero value for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) is only applied as the binary log expiration period if [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is not specified or is specified as 0.

To disable automatic purging of the binary log, specify a value of 0 explicitly for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds), and do not specify a value for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days). For compatibility with earlier releases, automatic purging is also disabled if you specify a value of 0 explicitly for [**expire\_logs\_days**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_expire_logs_days) and do not specify a value for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds). In that case, the default for [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) is not applied.

To remove binary log files manually, use the [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) statement. See [Section 13.4.1.1, “PURGE BINARY LOGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs).

**[log\_bin](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_bin)**

|  |  |
| --- | --- |
| **System Variable** | [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |

Shows the status of binary logging on the server, either enabled (**ON**) or disabled (**OFF**). With binary logging enabled, the server logs all statements that change data to the binary log, which is used for backup and replication. **ON** means that the binary log is available, **OFF** means that it is not in use. The [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option can be used to specify a base name and location for the binary log.

In earlier MySQL versions, binary logging was disabled by default, and was enabled if you specified the --log-bin option. From MySQL 8.0, binary logging is enabled by default, with the **log\_bin** system variable set to **ON**, whether or not you specify the --log-bin option. The exception is if you use [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) to initialize the data directory manually by invoking it with the --initialize or --initialize-insecure option, when binary logging is disabled by default. It is possible to enable binary logging in this case by specifying the --log-bin option.

If the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) or [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option is specified at startup, binary logging is disabled, with the **log\_bin** system variable set to **OFF**. If either of these options is specified and --log-bin is also specified, the option specified later takes precedence.

For information on the format and management of the binary log, see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

**[log\_bin\_basename](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_bin_basename)**

|  |  |
| --- | --- |
| **System Variable** | [**log\_bin\_basename**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_basename) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |

Holds the base name and path for the binary log files, which can be set with the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) server option. The maximum variable length is 256. In MySQL 8.0, if the --log-bin option is not supplied, the default base name is binlog. For compatibility with MySQL 5.7, if the --log-bin option is supplied with no string or with an empty string, the default base name is ***host\_name***-bin, using the name of the host machine. The default location is the data directory.

**[log\_bin\_index](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_bin_index)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-bin-index=file\_name** |
| **System Variable** | [**log\_bin\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_index) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | File name |

Holds the base name and path for the binary log index file, which can be set with the [--log-bin-index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin-index) server option. The maximum variable length is 256.

**[log\_bin\_trust\_function\_creators](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_bin_trust_function_creators)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-bin-trust-function-creators[={OFF|ON}]** |
| **System Variable** | [**log\_bin\_trust\_function\_creators**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_trust_function_creators) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This variable applies when binary logging is enabled. It controls whether stored function creators can be trusted not to create stored functions that may cause unsafe events to be written to the binary log. If set to 0 (the default), users are not permitted to create or alter stored functions unless they have the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege in addition to the [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine) or [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) privilege. A setting of 0 also enforces the restriction that a function must be declared with the **DETERMINISTIC** characteristic, or with the **READS SQL DATA** or **NO SQL** characteristic. If the variable is set to 1, MySQL does not enforce these restrictions on stored function creation. This variable also applies to trigger creation. See [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

**[log\_bin\_use\_v1\_row\_events](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_bin_use_v1_row_events)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-bin-use-v1-row-events[={OFF|ON}]** |
| **Deprecated** | 8.0.18 |
| **System Variable** | [**log\_bin\_use\_v1\_row\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin_use_v1_row_events) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

This read-only system variable is deprecated. Setting the system variable to **ON** at server startup enabled row-based replication with replicas running MySQL Server 5.5 and earlier by writing the binary log using Version 1 binary log row events, instead of Version 2 binary log row events which are the default as of MySQL 5.6.

**[log\_slave\_updates](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_slave_updates)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-slave-updates[={OFF|ON}]** |
| **System Variable** | [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

Whether updates received by a replica server from a replication source server should be logged to the replica's own binary log.

Enabling this variable causes the replica to write the updates that are received from a source and performed by the replication SQL thread to the replica's own binary log. Binary logging, which is controlled by the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option and is enabled by default, must also be enabled on the replica for updates to be logged. See [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options). [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) is enabled by default, unless you specify [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) to disable binary logging, in which case MySQL also disables replica update logging by default. If you need to disable replica update logging when binary logging is enabled, specify [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) at replica server startup.

Enabling [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) enables replication servers to be chained. For example, you might want to set up replication servers using this arrangement:

A -> B -> C

Here, **A** serves as the source for the replica **B**, and **B** serves as the source for the replica **C**. For this to work, **B** must be both a source *and* a replica. With binary logging enabled and [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) enabled, which are the default settings, updates received from **A** are logged by **B** to its binary log, and can therefore be passed on to **C**.

**[log\_statements\_unsafe\_for\_binlog](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_log_statements_unsafe_for_binlog)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--log-statements-unsafe-for-binlog[={OFF|ON}]** |
| **System Variable** | [**log\_statements\_unsafe\_for\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_statements_unsafe_for_binlog) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

If error 1592 is encountered, controls whether the generated warnings are added to the error log or not.

**[master\_verify\_checksum](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_master_verify_checksum)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--master-verify-checksum[={OFF|ON}]** |
| **System Variable** | [**master\_verify\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_verify_checksum) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **OFF** |

Enabling this variable causes the source to verify events read from the binary log by examining checksums, and to stop with an error in the event of a mismatch. [**master\_verify\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_verify_checksum) is disabled by default; in this case, the source uses the event length from the binary log to verify events, so that only complete events are read from the binary log.

**[max\_binlog\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_max_binlog_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-binlog-cache-size=#** |
| **System Variable** | [**max\_binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_cache_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **18446744073709547520** |
| **Minimum Value** | **4096** |
| **Maximum Value** | **18446744073709547520** |

If a transaction requires more than this many bytes of memory, the server generates a Multi-statement transaction required more than 'max\_binlog\_cache\_size' bytes of storage error. The minimum value is 4096. The maximum possible value is 16EiB (exbibytes). The maximum recommended value is 4GB; this is due to the fact that MySQL currently cannot work with binary log positions greater than 4GB. The value must be a multiple of 4096.

**max\_binlog\_cache\_size** sets the size for the transaction cache only; the upper limit for the statement cache is governed by the [**max\_binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_stmt_cache_size) system variable.

The visibility to sessions of **max\_binlog\_cache\_size** matches that of the [**binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_cache_size) system variable; in other words, changing its value affects only new sessions that are started after the value is changed.

**[max\_binlog\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_max_binlog_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-binlog-size=#** |
| **System Variable** | [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1073741824** |
| **Minimum Value** | **4096** |
| **Maximum Value** | **1073741824** |

If a write to the binary log causes the current log file size to exceed the value of this variable, the server rotates the binary logs (closes the current file and opens the next one). The minimum value is 4096 bytes. The maximum and default value is 1GB. Encrypted binary log files have an additional 512-byte header, which is included in [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size).

A transaction is written in one chunk to the binary log, so it is never split between several binary logs. Therefore, if you have big transactions, you might see binary log files larger than [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size).

If [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) is 0, the value of [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) applies to relay logs as well.

With GTIDs in use on the server, when [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) is reached, if the system table **mysql.gtid\_executed** cannot be accessed to write the GTIDs from the current binary log file, the binary log cannot be rotated. In this situation, the server responds according to its [**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action) setting. If **IGNORE\_ERROR** is set, an error is logged on the server and binary logging is halted, or if **ABORT\_SERVER** is set, the server shuts down.

**[max\_binlog\_stmt\_cache\_size](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_max_binlog_stmt_cache_size)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--max-binlog-stmt-cache-size=#** |
| **System Variable** | [**max\_binlog\_stmt\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_stmt_cache_size) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **18446744073709547520** |
| **Minimum Value** | **4096** |
| **Maximum Value** | **18446744073709547520** |

If nontransactional statements within a transaction require more than this many bytes of memory, the server generates an error. The minimum value is 4096. The maximum and default values are 4GB on 32-bit platforms and 16EB (exabytes) on 64-bit platforms. The value must be a multiple of 4096.

**max\_binlog\_stmt\_cache\_size** sets the size for the statement cache only; the upper limit for the transaction cache is governed exclusively by the [**max\_binlog\_cache\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_cache_size) system variable.

**[original\_commit\_timestamp](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_original_commit_timestamp)**

|  |  |
| --- | --- |
| **System Variable** | [**original\_commit\_timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_commit_timestamp) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Numeric |

For internal use by replication. When re-executing a transaction on a replica, this is set to the time when the transaction was committed on the original source, measured in microseconds since the epoch. This allows the original commit timestamp to be propagated throughout a replication topology.

Setting the session value of this system variable is a restricted operation. The session user must have either the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege (see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)), or privileges sufficient to set restricted session variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)). However, note that the variable is not intended for users to set; it is set automatically by the replication infrastructure.

**[sql\_log\_bin](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sql_log_bin)**

|  |  |
| --- | --- |
| **System Variable** | [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

This variable controls whether logging to the binary log is enabled for the current session (assuming that the binary log itself is enabled). The default value is **ON**. To disable or enable binary logging for the current session, set the session [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) variable to **OFF** or **ON**.

Set this variable to **OFF** for a session to temporarily disable binary logging while making changes to the source you do not want replicated to the replica.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

It is not possible to set the session value of [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) within a transaction or subquery.

*Setting this variable to****OFF****prevents GTIDs from being assigned to transactions in the binary log*. If you are using GTIDs for replication, this means that even when binary logging is later enabled again, the GTIDs written into the log from this point do not account for any transactions that occurred in the meantime, so in effect those transactions are lost.

**[sync\_binlog](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_sync_binlog)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--sync-binlog=#** |
| **System Variable** | [**sync\_binlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value** | **1** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

Controls how often the MySQL server synchronizes the binary log to disk.

[**sync\_binlog=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog): Disables synchronization of the binary log to disk by the MySQL server. Instead, the MySQL server relies on the operating system to flush the binary log to disk from time to time as it does for any other file. This setting provides the best performance, but in the event of a power failure or operating system crash, it is possible that the server has committed transactions that have not been synchronized to the binary log.

[**sync\_binlog=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog): Enables synchronization of the binary log to disk before transactions are committed. This is the safest setting but can have a negative impact on performance due to the increased number of disk writes. In the event of a power failure or operating system crash, transactions that are missing from the binary log are only in a prepared state. This permits the automatic recovery routine to roll back the transactions, which guarantees that no transaction is lost from the binary log.

[**sync\_binlog=*N***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog), where ***N*** is a value other than 0 or 1: The binary log is synchronized to disk after **N** binary log commit groups have been collected. In the event of a power failure or operating system crash, it is possible that the server has committed transactions that have not been flushed to the binary log. This setting can have a negative impact on performance due to the increased number of disk writes. A higher value improves performance, but with an increased risk of data loss.

For the greatest possible durability and consistency in a replication setup that uses **InnoDB** with transactions, use these settings:

[**sync\_binlog=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog).

[**innodb\_flush\_log\_at\_trx\_commit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit).

**Caution**

Many operating systems and some disk hardware fool the flush-to-disk operation. They may tell [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) that the flush has taken place, even though it has not. In this case, the durability of transactions is not guaranteed even with the recommended settings, and in the worst case, a power outage can corrupt **InnoDB** data. Using a battery-backed disk cache in the SCSI disk controller or in the disk itself speeds up file flushes, and makes the operation safer. You can also try to disable the caching of disk writes in hardware caches.

**[transaction\_write\_set\_extraction](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_transaction_write_set_extraction)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--transaction-write-set-extraction[=value]** |
| **System Variable** | [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) |
| **Scope** | Global, Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **XXHASH64** |
| **Valid Values** | **OFF**  **MURMUR32**  **XXHASH64** |

For a replication source server that has multithreaded replicas (replicas on which [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is set to a value greater than 0), where [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) is set to **WRITESET** or **WRITESET\_SESSION** to generate dependency information from the source's write set, [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) specifies the algorithm used to hash the writes extracted during a transaction. [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) must be set to **ROW** to change the value of this system variable.

When **WRITESET** or **WRITESET\_SESSION** is set as the value for [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking), [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) must be set to specify an algorithm (not set to **OFF**). The default in MySQL 8.0 is that [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) is set to **XXHASH64**. While the current value of [**binlog\_transaction\_dependency\_tracking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_dependency_tracking) is **WRITESET** or **WRITESET\_SESSION**, you cannot change the value of [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction).

For Group Replication, [**transaction\_write\_set\_extraction**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_transaction_write_set_extraction) must be set to **XXHASH64**. The process of extracting the writes from a transaction is used in Group Replication for conflict detection and certification on all group members. See [Section 18.3.1, “Group Replication Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-requirements).

As of MySQL 8.0.14, setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

**17.1.6.5 Global Transaction ID System Variables**

The MySQL Server system variables described in this section are used to monitor and control Global Transaction Identifiers (GTIDs). For additional information, see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids).

**[binlog\_gtid\_simple\_recovery](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_binlog_gtid_simple_recovery)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--binlog-gtid-simple-recovery[={OFF|ON}]** |
| **System Variable** | [**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) |
| **Scope** | Global |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Boolean |
| **Default Value** | **ON** |

This variable controls how binary log files are iterated during the search for GTIDs when MySQL starts or restarts.

When [**binlog\_gtid\_simple\_recovery=TRUE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery), which is the default in MySQL 8.0, the values of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) are computed at startup based on the values of **Previous\_gtids\_log\_event** in the most recent and oldest binary log files. For a description of the computation, see [The **gtid\_purged** System Variable](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-purged). This setting accesses only two binary log files during server restart. If all binary logs on the server were generated using MySQL 5.7.8 or later, [**binlog\_gtid\_simple\_recovery=TRUE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) can always safely be used.

If any binary logs from MySQL 5.7.7 or older are present on the server (for example, following an upgrade of an older server to MySQL 8.0), with [**binlog\_gtid\_simple\_recovery=TRUE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery), [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) might be initialized incorrectly in the following two situations:

The newest binary log was generated by MySQL 5.7.5 or earlier, and [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) was **ON** for some binary logs but **OFF** for the newest binary log.

A **SET @@GLOBAL.gtid\_purged** statement was issued on MySQL 5.7.7 or earlier, and the binary log that was active at the time of the **SET @@GLOBAL.gtid\_purged** statement has not yet been purged.

If an incorrect GTID set is computed in either situation, it remains incorrect even if the server is later restarted with [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery). If either of these situations apply or might apply on the server, set [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) before starting or restarting the server.

When [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) is set, the method of computing [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) as described in [The **gtid\_purged** System Variable](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-purged) is changed to iterate the binary log files as follows:

Instead of using the value of **Previous\_gtids\_log\_event** and GTID log events from the newest binary log file, the computation for [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) iterates from the newest binary log file, and uses the value of **Previous\_gtids\_log\_event** and any GTID log events from the first binary log file where it finds a **Previous\_gtids\_log\_event** value. If the server's most recent binary log files do not have GTID log events, for example if [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) was used but the server was later changed to [**gtid\_mode=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), this process can take a long time.

Instead of using the value of **Previous\_gtids\_log\_event** from the oldest binary log file, the computation for [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) iterates from the oldest binary log file, and uses the value of **Previous\_gtids\_log\_event** from the first binary log file where it finds either a nonempty **Previous\_gtids\_log\_event** value, or at least one GTID log event (indicating that the use of GTIDs starts at that point). If the server's older binary log files do not have GTID log events, for example if [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) was only set recently on the server, this process can take a long time.

**[enforce\_gtid\_consistency](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_enforce_gtid_consistency)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--enforce-gtid-consistency[=value]** |
| **System Variable** | [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **OFF** |
| **Valid Values** | **OFF**  **ON**  **WARN** |

Depending on the value of this variable, the server enforces GTID consistency by allowing execution of only statements that can be safely logged using a GTID. You *must* set this variable to **ON** before enabling GTID based replication.

The values that [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) can be configured to are:

**OFF**: all transactions are allowed to violate GTID consistency.

**ON**: no transaction is allowed to violate GTID consistency.

**WARN**: all transactions are allowed to violate GTID consistency, but a warning is generated in this case.

[--enforce-gtid-consistency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) only takes effect if binary logging takes place for a statement. If binary logging is disabled on the server, or if statements are not written to the binary log because they are removed by a filter, GTID consistency is not checked or enforced for the statements that are not logged.

Only statements that can be logged using GTID safe statements can be logged when [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) is set to **ON**, so the operations listed here cannot be used with this option:

[**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**DROP TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements inside transactions.

Transactions or statements that update both transactional and nontransactional tables. There is an exception that nontransactional DML is allowed in the same transaction or in the same statement as transactional DML, if all *nontransactional* tables are temporary.

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements, prior to MySQL 8.0.21. From MySQL 8.0.21, [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements are allowed for storage engines that support atomic DDL.

For more information, see [Section 17.1.3.7, “Restrictions on Replication with GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restrictions).

Prior to MySQL 5.7 and in early releases in that release series, the boolean [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) defaulted to **OFF**. To maintain compatibility with these earlier releases, the enumeration defaults to **OFF**, and setting [--enforce-gtid-consistency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) without a value is interpreted as setting the value to **ON**. The variable also has multiple textual aliases for the values: **0=OFF=FALSE**, **1=ON=TRUE**,**2=WARN**. This differs from other enumeration types but maintains compatibility with the boolean type used in previous releases. These changes impact on what is returned by the variable. Using **SELECT @@ENFORCE\_GTID\_CONSISTENCY**, **SHOW VARIABLES LIKE 'ENFORCE\_GTID\_CONSISTENCY'**, and **SELECT \* FROM INFORMATION\_SCHEMA.VARIABLES WHERE 'VARIABLE\_NAME' = 'ENFORCE\_GTID\_CONSISTENCY'**, all return the textual form, not the numeric form. This is an incompatible change, since **@@ENFORCE\_GTID\_CONSISTENCY** returns the numeric form for booleans but returns the textual form for **SHOW** and the Information Schema.

**[gtid\_executed](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_executed)**

|  |  |
| --- | --- |
| **System Variable** | [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) |
| **System Variable** | [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) |
| **Scope** | Global |
| **Scope** | Global, Session |
| **Dynamic** | No |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Unit** | **set of GTIDs** |

When used with global scope, this variable contains a representation of the set of all transactions executed on the server and GTIDs that have been set by a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) statement. This is the same as the value of the **Executed\_Gtid\_Set** column in the output of [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) and [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). The value of this variable is a GTID set, see [GTID Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts-gtid-sets) for more information.

When the server starts, **@@GLOBAL.gtid\_executed** is initialized. See [**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) for more information on how binary logs are iterated to populate [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). GTIDs are then added to the set as transactions are executed, or if any [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) statement is executed.

The set of transactions that can be found in the binary logs at any given time is equal to [**GTID\_SUBTRACT(@@GLOBAL.gtid\_executed, @@GLOBAL.gtid\_purged)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_gtid-subtract); that is, to all transactions in the binary log that have not yet been purged.

Issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) causes the global value (but not the session value) of this variable to be reset to an empty string. GTIDs are not otherwise removed from this set other than when the set is cleared due to **RESET MASTER**.

In some older releases, this variable could also be used with session scope, where it contained a representation of the set of transactions that are written to the cache in the current session. The session scope is now deprecated.

**[gtid\_executed\_compression\_period](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_executed_compression_period)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--gtid-executed-compression-period=#** |
| **System Variable** | [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Integer |
| **Default Value (≥ 8.0.23)** | **0** |
| **Default Value (≤ 8.0.22)** | **1000** |
| **Minimum Value** | **0** |
| **Maximum Value** | **4294967295** |

Compress the **mysql.gtid\_executed** table each time this many transactions have been processed. When binary logging is enabled on the server, this compression method is not used, and instead the **mysql.gtid\_executed** table is compressed on each binary log rotation. When binary logging is disabled on the server, the compression thread sleeps until the specified number of transactions have been executed, then wakes up to perform compression of the **mysql.gtid\_executed** table. Setting the value of this system variable to 0 means that the thread never wakes up, so this explicit compression method is not used. Instead, compression occurs implicitly as required.

From MySQL 8.0.17, **InnoDB** transactions are written to the **mysql.gtid\_executed** table by a separate process to non-**InnoDB** transactions. If the server has a mix of **InnoDB** transactions and non-**InnoDB** transactions, the compression controlled by this system variable interferes with the work of this process and can slow it significantly. For this reason, from that release it is recommended that you set [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) to 0.

From MySQL 8.0.23, **InnoDB** and non-**InnoDB** transactions are written to the **mysql.gtid\_executed** table by the same process, and the [**gtid\_executed\_compression\_period**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed_compression_period) default value is 0.

See [mysql.gtid\_executed Table Compression](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-executed-table-compression) for more information.

**[gtid\_mode](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_mode)**

|  |  |
| --- | --- |
| **Command-Line Format** | **--gtid-mode=MODE** |
| **System Variable** | [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **OFF** |
| **Valid Values** | **OFF**  **OFF\_PERMISSIVE**  **ON\_PERMISSIVE**  **ON** |

Controls whether GTID based logging is enabled and what type of transactions the logs can contain. You must have privileges sufficient to set global system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges). [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) must be set to **ON** before you can set [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode). Before modifying this variable, see [Section 17.1.4, “Changing GTID Mode on Online Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online).

Logged transactions can be either anonymous or use GTIDs. Anonymous transactions rely on binary log file and position to identify specific transactions. GTID transactions have a unique identifier that is used to refer to transactions. The different modes are:

**OFF**: Both new and replicated transactions must be anonymous.

**OFF\_PERMISSIVE**: New transactions are anonymous. Replicated transactions can be either anonymous or GTID transactions.

**ON\_PERMISSIVE**: New transactions are GTID transactions. Replicated transactions can be either anonymous or GTID transactions.

**ON**: Both new and replicated transactions must be GTID transactions.

Changes from one value to another can only be one step at a time. For example, if [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is currently set to **OFF\_PERMISSIVE**, it is possible to change to **OFF** or **ON\_PERMISSIVE** but not to **ON**.

The values of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) and [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) are persistent regardless of the value of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode). Therefore even after changing the value of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), these variables contain the correct values.

**[gtid\_next](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_next)**

|  |  |
| --- | --- |
| **System Variable** | [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) |
| **Scope** | Session |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | Enumeration |
| **Default Value** | **AUTOMATIC** |
| **Valid Values** | **AUTOMATIC**  **ANONYMOUS**  **UUID:NUMBER** |

This variable is used to specify whether and how the next GTID is obtained.

Setting the session value of this system variable is a restricted operation. The session user must have either the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege (see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)), or privileges sufficient to set restricted session variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)).

**gtid\_next** can take any of the following values:

**AUTOMATIC**: Use the next automatically-generated global transaction ID.

**ANONYMOUS**: Transactions do not have global identifiers, and are identified by file and position only.

A global transaction ID in ***UUID***:***NUMBER*** format.

Exactly which of the above options are valid depends on the setting of [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), see [Section 17.1.4.1, “Replication Mode Concepts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-mode-change-online-concepts) for more information. Setting this variable has no effect if [**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **OFF**.

After this variable has been set to ***UUID***:***NUMBER***, and a transaction has been committed or rolled back, an explicit **SET GTID\_NEXT** statement must again be issued before any other statement.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) or [**DROP TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) fails with an explicit error when used on a combination of nontemporary tables with temporary tables, or of temporary tables using transactional storage engines with temporary tables using nontransactional storage engines.

**[gtid\_owned](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_owned)**

|  |  |
| --- | --- |
| **System Variable** | [**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned) |
| **Scope** | Global, Session |
| **Dynamic** | No |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Unit** | **set of GTIDs** |

This read-only variable is primarily for internal use. Its contents depend on its scope.

When used with global scope, [**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned) holds a list of all the GTIDs that are currently in use on the server, with the IDs of the threads that own them. This variable is mainly useful for a multi-threaded replica to check whether a transaction is already being applied on another thread. An applier thread takes ownership of a transaction's GTID all the time it is processing the transaction, so **@@global.gtid\_owned** shows the GTID and owner for the duration of processing. When a transaction has been committed (or rolled back), the applier thread releases ownership of the GTID.

When used with session scope, [**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned) holds a single GTID that is currently in use by and owned by this session. This variable is mainly useful for testing and debugging the use of GTIDs when the client has explicitly assigned a GTID for the transaction by setting [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next). In this case, **@@session.gtid\_owned** displays the GTID all the time the client is processing the transaction, until the transaction has been committed (or rolled back). When the client has finished processing the transaction, the variable is cleared. If [**gtid\_next=AUTOMATIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) is used for the session, [**gtid\_owned**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_owned) is populated only briefly during the execution of the commit statement for the transaction, so it cannot be observed from the session concerned, although it is listed if **@@global.gtid\_owned** is read at the right point. If you have a requirement to track the GTIDs that are handled by a client in a session, you can enable the session state tracker controlled by the [**session\_track\_gtids**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_session_track_gtids) system variable.

**[gtid\_purged](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\replication.html" \l "sysvar_gtid_purged)**

|  |  |
| --- | --- |
| **System Variable** | [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) |
| **Scope** | Global |
| **Dynamic** | Yes |
| [**SET\_VAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-set-var)**Hint Applies** | No |
| **Type** | String |
| **Unit** | **set of GTIDs** |

The global value of the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable (**@@GLOBAL.gtid\_purged**) is a GTID set consisting of the GTIDs of all the transactions that have been committed on the server, but do not exist in any binary log file on the server. [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) is a subset of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). The following categories of GTIDs are in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged):

GTIDs of replicated transactions that were committed with binary logging disabled on the replica.

GTIDs of transactions that were written to a binary log file that has now been purged.

GTIDs that were added explicitly to the set by the statement **SET @@GLOBAL.gtid\_purged**.

When the server starts, the global value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) is initialized to a set of GTIDs. For information on how this GTID set is computed, see [The **gtid\_purged** System Variable](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-purged). If binary logs from MySQL 5.7.7 or older are present on the server, you might need to set [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) in the server's configuration file to produce the correct computation. See the description for [**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) for details of the situations in which this setting is needed.

Issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) causes the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) to be reset to an empty string.

You can set the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) in order to record on the server that the transactions in a certain GTID set have been applied, although they do not exist in any binary log on the server. An example use case for this action is when you are restoring a backup of one or more databases on a server, but you do not have the relevant binary logs containing the transactions on the server.

**Important**

GTIDs are only available on a server instance up to the number of non-negative values for a signed 64-bit integer (2 to the power of 63, minus 1). If you set the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) to a number that approaches this limit, subsequent commits can cause the server to run out of GTIDs and take the action specified by [**binlog\_error\_action**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_error_action). From MySQL 8.0.23, a warning message is issued when the server instance is approaching the limit.

From MySQL 8.0, there are two ways to set the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged). You can either replace the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) with your specified GTID set, or you can append your specified GTID set to the GTID set that is already held by [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged). If the server has no existing GTIDs, for example an empty server that you are provisioning with a backup of an existing database, both methods have the same result. If you are restoring a backup that overlaps the transactions that are already on the server, for example replacing a corrupted table with a partial dump from the source made using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) (which includes the GTIDs of all the transactions on the server, even though the dump is partial), use the first method of replacing the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged). If you are restoring a backup that is disjoint from the transactions that are already on the server, for example provisioning a multi-source replica using dumps from two different servers, use the second method of adding to the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged).

To replace the value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) with your specified GTID set, use the following statement:

SET @@GLOBAL.gtid\_purged = 'gtid\_set'

**gtid\_set** must be a superset of the current value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), and must not intersect with **gtid\_subtract(gtid\_executed,gtid\_purged)**. In other words, the new GTID set ***must*** include any GTIDs that were already in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), and ***must not*** include any GTIDs in [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) that have not yet been purged. **gtid\_set** also cannot include any GTIDs that are in **@@global.gtid\_owned**, that is, the GTIDs for transactions that are currently being processed on the server.

The result is that the global value of [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) is set equal to **gtid\_set**, and the value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) becomes the union of **gtid\_set** and the previous value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed).

To append your specified GTID set to [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged), use the following statement with a plus sign (+) before the GTID set:

SET @@GLOBAL.gtid\_purged = '+gtid\_set'

**gtid\_set** ***must not*** intersect with the current value of [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed). In other words, the new GTID set must not include any GTIDs in [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed), including transactions that are already also in [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged). **gtid\_set** also cannot include any GTIDs that are in **@@global.gtid\_owned**, that is, the GTIDs for transactions that are currently being processed on the server.

The result is that **gtid\_set** is added to both [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) and [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged).

**Note**

If any binary logs from MySQL 5.7.7 or older are present on the server (for example, following an upgrade of an older server to MySQL 8.0), after issuing a **SET @@GLOBAL.gtid\_purged** statement, you might need to set [**binlog\_gtid\_simple\_recovery=FALSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) in the server's configuration file before restarting the server, otherwise [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) can be computed incorrectly. See the description for [**binlog\_gtid\_simple\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_gtid_simple_recovery) for details of the situations in which this setting is needed.

**17.1.7 Common Replication Administration Tasks**

[17.1.7.1 Checking Replication Status](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-status)

[17.1.7.2 Pausing Replication on the Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-pausing)

[17.1.7.3 Skipping Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip)

Once replication has been started it executes without requiring much regular administration. This section describes how to check the status of replication, how to pause a replica, and how to skip a failed transaction on a replica.

Tip

To deploy multiple instances of MySQL, you can use [InnoDB Cluster](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html) which enables you to easily administer a group of MySQL server instances in [MySQL Shell](https://dev.mysql.com/doc/mysql-shell/8.0/en/). InnoDB Cluster wraps MySQL Group Replication in a programmatic environment that enables you easily deploy a cluster of MySQL instances to achieve high availability. In addition, InnoDB Cluster interfaces seamlessly with [MySQL Router](https://dev.mysql.com/doc/mysql-router/8.0/en/), which enables your applications to connect to the cluster without writing your own failover process. For similar use cases that do not require high availability, however, you can use [InnoDB ReplicaSet](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

**17.1.7.1 Checking Replication Status**

The most common task when managing a replication process is to ensure that replication is taking place and that there have been no errors between the replica and the source.

The [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement, which you must execute on each replica, provides information about the configuration and status of the connection between the replica server and the source server. From MySQL 8.0.22, [**SHOW SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status) is deprecated, and [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) is available to use instead. The Performance Schema has replication tables that provide this information in a more accessible form. See [Section 27.12.11, “Performance Schema Replication Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-tables).

The replication heartbeat information shown in the Performance Schema replication tables lets you check that the replication connection is active even if the source has not sent events to the replica recently. The source sends a heartbeat signal to a replica if there are no updates to, and no unsent events in, the binary log for a longer period than the heartbeat interval. The **MASTER\_HEARTBEAT\_PERIOD** setting on the source (set by the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement) specifies the frequency of the heartbeat, which defaults to half of the connection timeout interval for the replica ([**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout)). The [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) Performance Schema table shows when the most recent heartbeat signal was received by a replica, and how many heartbeat signals it has received.

If you are using the [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement to check on the status of an individual replica, the statement provides the following information:

mysql> **SHOW REPLICA STATUS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Replica\_IO\_State: Waiting for master to send event

Source\_Host: source1

Source\_User: root

Source\_Port: 3306

Connect\_Retry: 60

Source\_Log\_File: mysql-bin.000004

Read\_Source\_Log\_Pos: 931

Relay\_Log\_File: replica1-relay-bin.000056

Relay\_Log\_Pos: 950

Relay\_Source\_Log\_File: mysql-bin.000004

Replica\_IO\_Running: Yes

Replica\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno: 0

Last\_Error:

Skip\_Counter: 0

Exec\_Source\_Log\_Pos: 931

Relay\_Log\_Space: 1365

Until\_Condition: None

Until\_Log\_File:

Until\_Log\_Pos: 0

Source\_SSL\_Allowed: No

Source\_SSL\_CA\_File:

Source\_SSL\_CA\_Path:

Source\_SSL\_Cert:

Source\_SSL\_Cipher:

Source\_SSL\_Key:

Seconds\_Behind\_Source: 0

Source\_SSL\_Verify\_Server\_Cert: No

Last\_IO\_Errno: 0

Last\_IO\_Error:

Last\_SQL\_Errno: 0

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids: 0

The key fields from the status report to examine are:

**Replica\_IO\_State**: The current status of the replica. See [Section 8.14.5, “Replication I/O Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-io-thread-states), and [Section 8.14.6, “Replication SQL Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-sql-thread-states), for more information.

**Replica\_IO\_Running**: Whether the I/O thread for reading the source's binary log is running. Normally, you want this to be **Yes** unless you have not yet started replication or have explicitly stopped it with [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica).

**Replica\_SQL\_Running**: Whether the SQL thread for executing events in the relay log is running. As with the I/O thread, this should normally be **Yes**.

**Last\_IO\_Error**, **Last\_SQL\_Error**: The last errors registered by the I/O and SQL threads when processing the relay log. Ideally these should be blank, indicating no errors.

**Seconds\_Behind\_Source**: The number of seconds that the replication SQL thread is behind processing the source binary log. A high number (or an increasing one) can indicate that the replica is unable to handle events from the source in a timely fashion.

A value of 0 for **Seconds\_Behind\_Source** can usually be interpreted as meaning that the replica has caught up with the source, but there are some cases where this is not strictly true. For example, this can occur if the network connection between source and replica is broken but the replication I/O thread has not yet noticed this; that is, [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) has not yet elapsed.

It is also possible that transient values for **Seconds\_Behind\_Source** may not reflect the situation accurately. When the replication SQL thread has caught up on I/O, **Seconds\_Behind\_Source** displays 0; but when the replication I/O thread is still queuing up a new event, **Seconds\_Behind\_Source** may show a large value until the replication SQL thread finishes executing the new event. This is especially likely when the events have old timestamps; in such cases, if you execute [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) several times in a relatively short period, you may see this value change back and forth repeatedly between 0 and a relatively large value.

Several pairs of fields provide information about the progress of the replica in reading events from the source binary log and processing them in the relay log:

(**Master\_Log\_file**, **Read\_Master\_Log\_Pos**): Coordinates in the source binary log indicating how far the replication I/O thread has read events from that log.

(**Relay\_Master\_Log\_File**, **Exec\_Master\_Log\_Pos**): Coordinates in the source binary log indicating how far the replication SQL thread has executed events received from that log.

(**Relay\_Log\_File**, **Relay\_Log\_Pos**): Coordinates in the replica relay log indicating how far the replication SQL thread has executed the relay log. These correspond to the preceding coordinates, but are expressed in replica relay log coordinates rather than source binary log coordinates.

On the source, you can check the status of connected replicas using [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) to examine the list of running processes. Replica connections have **Binlog Dump** in the **Command** field:

mysql> **SHOW PROCESSLIST \G;**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 10

User: root

Host: replica1:58371

db: NULL

Command: Binlog Dump

Time: 777

State: Has sent all binlog to slave; waiting for binlog to be updated

Info: NULL

Because it is the replica that drives the replication process, very little information is available in this report.

For replicas that were started with the [--report-host](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) option and are connected to the source, the [**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) statement on the source shows basic information about the replicas. The output includes the ID of the replica server, the value of the [--report-host](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) option, the connecting port, and source ID:

mysql> **SHOW REPLICAS;**

+-----------+----------+------+-------------------+-----------+

| Server\_id | Host | Port | Rpl\_recovery\_rank | Source\_id |

+-----------+----------+------+-------------------+-----------+

| 10 | replica1 | 3306 | 0 | 1 |

+-----------+----------+------+-------------------+-----------+

1 row in set (0.00 sec)

**17.1.7.2 Pausing Replication on the Replica**

You can stop and start replication on the replica using the [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements. From MySQL 8.0.22, [**STOP SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-slave) and [**START SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave) are deprecated, and [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) are available to use instead.

To stop processing of the binary log from the source, use [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica):

mysql> **STOP SLAVE;**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA;**

When replication is stopped, the replication I/O thread stops reading events from the source binary log and writing them to the relay log, and the SQL thread stops reading events from the relay log and executing them. You can pause the I/O or SQL thread individually by specifying the thread type:

mysql> **STOP SLAVE IO\_THREAD;**

mysql> **STOP SLAVE SQL\_THREAD;**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA IO\_THREAD;**

mysql> **STOP REPLICA SQL\_THREAD;**

To start execution again, use the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement:

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **START REPLICA;**

To start a particular thread, specify the thread type:

mysql> **START SLAVE IO\_THREAD;**

mysql> **START SLAVE SQL\_THREAD;**

Or from MySQL 8.0.22:

mysql> **START REPLICA IO\_THREAD;**

mysql> **START REPLICA SQL\_THREAD;**

For a replica that performs updates only by processing events from the source, stopping only the SQL thread can be useful if you want to perform a backup or other task. The I/O thread continues to read events from the source but they are not executed. This makes it easier for the replica to catch up when you restart the SQL thread.

Stopping only the I/O thread enables the events in the relay log to be executed by the SQL thread up to the point where the relay log ends. This can be useful when you want to pause execution to catch up with events already received from the source, when you want to perform administration on the replica but also ensure that it has processed all updates to a specific point. This method can also be used to pause event receipt on the replica while you conduct administration on the source. Stopping the I/O thread but permitting the SQL thread to run helps ensure that there is not a massive backlog of events to be executed when replication is started again.

**17.1.7.3 Skipping Transactions**

If replication stops due to an issue with an event in a replicated transaction, you can resume replication by skipping the failed transaction on the replica. Before skipping a transaction, ensure that the replication I/O thread is stopped as well as the SQL thread.

First you need to identify the replicated event that caused the error. Details of the error and the last successfully applied transaction are recorded in the Performance Schema table [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table). You can use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to retrieve and display the events that were logged around the time of the error. For instructions to do this, see [Section 7.5, “Point-in-Time (Incremental) Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html#point-in-time-recovery). Alternatively, you can issue [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) on the replica or [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) on the source.

Before skipping the transaction and restarting the replica, check these points:

Is the transaction that stopped replication from an unknown or untrusted source? If so, investigate the cause in case there are any security considerations that indicate the replica should not be restarted.

Does the transaction that stopped replication need to be applied on the replica? If so, either make the appropriate corrections and reapply the transaction, or manually reconcile the data on the replica.

Did the transaction that stopped replication need to be applied on the source? If not, undo the transaction manually on the server where it originally took place.

To skip the transaction, choose one of the following methods as appropriate:

When GTIDs are in use ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **ON**), see [Section 17.1.7.3.1, “Skipping Transactions With GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip-gtid) .

When GTIDs are not in use or are being phased in ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **OFF**, **OFF\_PERMISSIVE**, or **ON\_PERMISSIVE**), see [Section 17.1.7.3.2, “Skipping Transactions Without GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip-nogtid).

If you have enabled GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement, see [Section 17.1.7.3.2, “Skipping Transactions Without GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip-nogtid). Using **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on a replication channel is not the same as introducing GTID-based replication for the channel, and you cannot use the transaction skipping method for GTID-based replication with those channels.

To restart replication after skipping the transaction, issue [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), with the **FOR CHANNEL** clause if the replica is a multi-source replica.

**17.1.7.3.1 Skipping Transactions With GTIDs**

When GTIDs are in use ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **ON**), the GTID for a committed transaction is persisted on the replica even if the content of the transaction is filtered out. This feature prevents a replica from retrieving previously filtered transactions when it reconnects to the source using GTID auto-positioning. It can also be used to skip a transaction on the replica, by committing an empty transaction in place of the failing transaction.

This method of skipping transactions is not suitable when you have enabled GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement.

If the failing transaction generated an error in a worker thread, you can obtain its GTID directly from the **LAST\_SEEN\_TRANSACTION** field in the Performance Schema table [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table). To see what the transaction is, issue [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) on the replica or [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) on the source, and search the output for a transaction preceded by that GTID.

When you have assessed the failing transaction for any other appropriate actions as described previously (such as security considerations), to skip it, commit an empty transaction on the replica that has the same GTID as the failing transaction. For example:

SET GTID\_NEXT='aaa-bbb-ccc-ddd:N';

BEGIN;

COMMIT;

SET GTID\_NEXT='AUTOMATIC';

The presence of this empty transaction on the replica means that when you issue a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement to restart replication, the replica uses the auto-skip function to ignore the failing transaction, because it sees a transaction with that GTID has already been applied. If the replica is a multi-source replica, you do not need to specify the channel name when you commit the empty transaction, but you do need to specify the channel name when you issue [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica).

Note that if binary logging is in use on this replica, the empty transaction enters the replication stream if the replica becomes a source or primary in the future. If you need to avoid this possibility, consider flushing and purging the replica's binary logs, as in this example:

FLUSH LOGS;

PURGE BINARY LOGS TO 'binlog.000146';

The GTID of the empty transaction is persisted, but the transaction itself is removed by purging the binary log files.

**17.1.7.3.2 Skipping Transactions Without GTIDs**

To skip failing transactions when GTIDs are not in use or are being phased in ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **OFF**, **OFF\_PERMISSIVE**, or **ON\_PERMISSIVE**), you can skip a specified number of events by issuing a **SET GLOBAL sql\_slave\_skip\_counter** statement. Alternatively, you can skip past an event or events by issuing a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement to move the source binary log position forward.

These methods are also suitable when you have enabled GTID assignment on a replication channel using the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement.

When you use these methods, it is important to understand that you are not necessarily skipping a complete transaction, as is always the case with the GTID-based method described previously. These non-GTID-based methods are not aware of transactions as such, but instead operate on events. The binary log is organized as a sequence of groups known as event groups, and each event group consists of a sequence of events.

For transactional tables, an event group corresponds to a transaction.

For nontransactional tables, an event group corresponds to a single SQL statement.

A single transaction can contain changes to both transactional and nontransactional tables.

When you use a **SET GLOBAL sql\_slave\_skip\_counter** statement to skip events and the resulting position is in the middle of an event group, the replica continues to skip events until it reaches the end of the group. Execution then starts with the next event group. The [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement does not have this function, so you must be careful to identify the correct location to restart replication at the beginning of an event group. However, using [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) means you do not have to count the events that need to be skipped, as you do with a **SET GLOBAL sql\_slave\_skip\_counter**, and instead you can just specify the location to restart.

**17.1.7.3.2.1 Skipping Transactions With SET GLOBAL sql\_slave\_skip\_counter**

When you have assessed the failing transaction for any other appropriate actions as described previously (such as security considerations), count the number of events that you need to skip. One event normally corresponds to one SQL statement in the binary log, but note that statements that use **AUTO\_INCREMENT** or **LAST\_INSERT\_ID()** count as two events in the binary log. When binary log transaction compression is in use, a compressed transaction payload (**Transaction\_payload\_event**) is counted as a single counter value, so all the events inside it are skipped as a unit.

If you want to skip the complete transaction, you can count the events to the end of the transaction, or you can just skip the relevant event group. Remember that with **SET GLOBAL sql\_slave\_skip\_counter**, the replica continues to skip to the end of an event group. Make sure you do not skip too far forward and go into the next event group or transaction so that it is not also skipped.

Issue the **SET** statement as follows, where ***N*** is the number of events from the source to skip:

SET GLOBAL sql\_slave\_skip\_counter = ***N***

This statement cannot be issued if [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set, or if the replication I/O and SQL threads are running.

The **SET GLOBAL sql\_slave\_skip\_counter** statement has no immediate effect. When you issue the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement for the next time following this **SET** statement, the new value for the system variable [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) is applied, and the events are skipped. That [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement also automatically sets the value of the system variable back to 0. If the replica is a multi-source replica, when you issue that [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement, the **FOR CHANNEL** clause is required. Make sure that you name the correct channel, otherwise events are skipped on the wrong channel.

**17.1.7.3.2.2 Skipping Transactions With CHANGE MASTER TO**

When you have assessed the failing transaction for any other appropriate actions as described previously (such as security considerations), identify the coordinates (file and position) in the source's binary log that represent a suitable position to restart replication. This can be the start of the event group following the event that caused the issue, or the start of the next transaction. The replication I/O thread begins reading from the source at these coordinates the next time the thread starts, skipping the failing event. Make sure that you have identified the position accurately, because this statement does not take event groups into account.

Issue the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement as follows, where ***source\_log\_name*** is the binary log file that contains the restart position, and ***source\_log\_pos*** is the number representing the restart position as stated in the binary log file:

CHANGE MASTER TO MASTER\_LOG\_FILE='***source\_log\_name***', MASTER\_LOG\_POS=***source\_log\_pos***;

If the replica is a multi-source replica, you must use the **FOR CHANNEL** clause to name the appropriate channel on the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement.

This statement cannot be issued if **MASTER\_AUTO\_POSITION=1** is set, or if the replication I/O and SQL threads are running. If you need to use this method of skipping a transaction when **MASTER\_AUTO\_POSITION=1** is normally set, you can change the setting to **MASTER\_AUTO\_POSITION=1** while issuing the statement, then change it back again afterwards. For example:

CHANGE MASTER TO MASTER\_AUTO\_POSITION=0, MASTER\_LOG\_FILE='binlog.000145', MASTER\_LOG\_POS=235;

CHANGE MASTER TO MASTER\_AUTO\_POSITION=1;

**17.2 Replication Implementation**

[17.2.1 Replication Formats](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats)

[17.2.2 Replication Channels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels)

[17.2.3 Replication Threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation-details)

[17.2.4 Relay Log and Replication Metadata Repositories](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)

[17.2.5 How Servers Evaluate Replication Filtering Rules](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules)

Replication is based on the source server keeping track of all changes to its databases (updates, deletes, and so on) in its binary log. The binary log serves as a written record of all events that modify database structure or content (data) from the moment the server was started. Typically, [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements are not recorded because they modify neither database structure nor content.

Each replica that connects to the source requests a copy of the binary log. That is, it pulls the data from the source, rather than the source pushing the data to the replica. The replica also executes the events from the binary log that it receives. This has the effect of repeating the original changes just as they were made on the source. Tables are created or their structure modified, and data is inserted, deleted, and updated according to the changes that were originally made on the source.

Because each replica is independent, the replaying of the changes from the source's binary log occurs independently on each replica that is connected to the source. In addition, because each replica receives a copy of the binary log only by requesting it from the source, the replica is able to read and update the copy of the database at its own pace and can start and stop the replication process at will without affecting the ability to update to the latest database status on either the source or replica side.

For more information on the specifics of the replication implementation, see [Section 17.2.3, “Replication Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation-details).

Source servers and replicas report their status in respect of the replication process regularly so that you can monitor them. See [Section 8.14, “Examining Server Thread (Process) Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#thread-information), for descriptions of all replicated-related states.

The source's binary log is written to a local relay log on the replica before it is processed. The replica also records information about the current position with the source's binary log and the local relay log. See [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs).

Database changes are filtered on the replica according to a set of rules that are applied according to the various configuration options and variables that control event evaluation. For details on how these rules are applied, see [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules).

**17.2.1 Replication Formats**

[17.2.1.1 Advantages and Disadvantages of Statement-Based and Row-Based Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr)

[17.2.1.2 Usage of Row-Based Logging and Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-usage)

[17.2.1.3 Determination of Safe and Unsafe Statements in Binary Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-safe-unsafe)

Replication works because events written to the binary log are read from the source and then processed on the replica. The events are recorded within the binary log in different formats according to the type of event. The different replication formats used correspond to the binary logging format used when the events were recorded in the source's binary log. The correlation between binary logging formats and the terms used during replication are:

When using statement-based binary logging, the source writes SQL statements to the binary log. Replication of the source to the replica works by executing the SQL statements on the replica. This is called statement-based replication (which can be abbreviated as SBR), which corresponds to the MySQL statement-based binary logging format.

When using row-based logging, the source writes events to the binary log that indicate how individual table rows are changed. Replication of the source to the replica works by copying the events representing the changes to the table rows to the replica. This is called row-based replication (which can be abbreviated as RBR).

Row-based logging is the default method.

You can also configure MySQL to use a mix of both statement-based and row-based logging, depending on which is most appropriate for the change to be logged. This is called mixed-format logging. When using mixed-format logging, a statement-based log is used by default. Depending on certain statements, and also the storage engine being used, the log is automatically switched to row-based in particular cases. Replication using the mixed format is referred to as mixed-based replication or mixed-format replication. For more information, see [Section 5.4.4.3, “Mixed Binary Logging Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-mixed).

**NDB Cluster.** The default binary logging format in MySQL NDB Cluster 8.0 is **MIXED**. You should note that NDB Cluster Replication always uses row-based replication, and that the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine is incompatible with statement-based replication. See [Section 23.6.2, “General Requirements for NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-general), for more information.

When using **MIXED** format, the binary logging format is determined in part by the storage engine being used and the statement being executed. For more information on mixed-format logging and the rules governing the support of different logging formats, see [Section 5.4.4.3, “Mixed Binary Logging Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-mixed).

The logging format in a running MySQL server is controlled by setting the [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) server system variable. This variable can be set with session or global scope. The rules governing when and how the new setting takes effect are the same as for other MySQL server system variables. Setting the variable for the current session lasts only until the end of that session, and the change is not visible to other sessions. Setting the variable globally takes effect for clients that connect after the change, but not for any current client sessions, including the session where the variable setting was changed. To make the global system variable setting permanent so that it applies across server restarts, you must set it in an option file. For more information, see [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

There are conditions under which you cannot change the binary logging format at runtime or doing so causes replication to fail. See [Section 5.4.4.2, “Setting The Binary Log Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-setting).

Changing the global [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) value requires privileges sufficient to set global system variables. Changing the session [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) value requires privileges sufficient to set restricted session system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

The statement-based and row-based replication formats have different issues and limitations. For a comparison of their relative advantages and disadvantages, see [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

With statement-based replication, you may encounter issues with replicating stored routines or triggers. You can avoid these issues by using row-based replication instead. For more information, see [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

**17.2.1.1 Advantages and Disadvantages of Statement-Based and Row-Based Replication**

Each binary logging format has advantages and disadvantages. For most users, the mixed replication format should provide the best combination of data integrity and performance. If, however, you want to take advantage of the features specific to the statement-based or row-based replication format when performing certain tasks, you can use the information in this section, which provides a summary of their relative advantages and disadvantages, to determine which is best for your needs.

[Advantages of statement-based replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr-sbr-advantages)

[Disadvantages of statement-based replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr-sbr-disadvantages)

[Advantages of row-based replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr-rbr-advantages)

[Disadvantages of row-based replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr-rbr-disadvantages)

**Advantages of statement-based replication**

Proven technology.

Less data written to log files. When updates or deletes affect many rows, this results in *much* less storage space required for log files. This also means that taking and restoring from backups can be accomplished more quickly.

Log files contain all statements that made any changes, so they can be used to audit the database.

**Disadvantages of statement-based replication**

**Statements that are unsafe for SBR.** Not all statements which modify data (such as [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements) can be replicated using statement-based replication. Any nondeterministic behavior is difficult to replicate when using statement-based replication. Examples of such Data Modification Language (DML) statements include the following:

A statement that depends on a UDF or stored program that is nondeterministic, since the value returned by such a UDF or stored program or depends on factors other than the parameters supplied to it. (Row-based replication, however, simply replicates the value returned by the UDF or stored program, so its effect on table rows and data is the same on both the source and replica.) See [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked), for more information.

[**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements that use a **LIMIT** clause without an **ORDER BY** are nondeterministic. See [Section 17.5.1.18, “Replication and LIMIT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-limit).

Locking read statements ([**SELECT ... FOR UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**SELECT ... FOR SHARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)) that use **NOWAIT** or **SKIP LOCKED** options. See [Locking Read Concurrency with NOWAIT and SKIP LOCKED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads-nowait-skip-locked).

Deterministic UDFs must be applied on the replicas.

Statements using any of the following functions cannot be replicated properly using statement-based replication:

[**LOAD\_FILE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file)

[**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid), [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short)

[**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user)

[**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows)

[**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) (unless both the source and the replica are started with the [--sysdate-is-now](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_sysdate-is-now) option)

[**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock)

[**IS\_FREE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-free-lock)

[**IS\_USED\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-used-lock)

[**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait)

[**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand)

[**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock)

[**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep)

[**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version)

However, all other functions are replicated correctly using statement-based replication, including [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) and so forth.

For more information, see [Section 17.5.1.14, “Replication and System Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-functions).

Statements that cannot be replicated correctly using statement-based replication are logged with a warning like the one shown here:

[Warning] Statement is not safe to log in statement format.

A similar warning is also issued to the client in such cases. The client can display it using [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

[**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) requires a greater number of row-level locks than with row-based replication.

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements that require a table scan (because no index is used in the **WHERE** clause) must lock a greater number of rows than with row-based replication.

For [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html): An [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement that uses **AUTO\_INCREMENT** blocks other nonconflicting [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements.

For complex statements, the statement must be evaluated and executed on the replica before the rows are updated or inserted. With row-based replication, the replica only has to modify the affected rows, not execute the full statement.

If there is an error in evaluation on the replica, particularly when executing complex statements, statement-based replication may slowly increase the margin of error across the affected rows over time. See [Section 17.5.1.29, “Replica Errors During Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-slaveerrors).

Stored functions execute with the same [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) value as the calling statement. However, this is not true of stored procedures.

Deterministic UDFs must be applied on the replicas.

Table definitions must be (nearly) identical on source and replica. See [Section 17.5.1.9, “Replication with Differing Table Definitions on Source and Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-differing-tables), for more information.

As of MySQL 8.0.22, DML operations that read data from MySQL grant tables (through a join list or subquery) but do not modify them are performed as non-locking reads on the MySQL grant tables and are therefore not safe for statement-based replication. For more information, see [Grant Table Concurrency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#grant-tables-concurrency).

**Advantages of row-based replication**

All changes can be replicated. This is the safest form of replication.

**Note**

Statements that update the information in the **mysql** system schema, such as [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) and the manipulation of triggers, stored routines (including stored procedures), and views, are all replicated to replicas using statement-based replication.

For statements such as [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), a **CREATE** statement is generated from the table definition and replicated using statement-based format, while the row insertions are replicated using row-based format.

Fewer row locks are required on the source, which thus achieves higher concurrency, for the following types of statements:

[**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select)

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements with **AUTO\_INCREMENT**

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements with **WHERE** clauses that do not use keys or do not change most of the examined rows.

Fewer row locks are required on the replica for any [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement.

**Disadvantages of row-based replication**

RBR can generate more data that must be logged. To replicate a DML statement (such as an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement), statement-based replication writes only the statement to the binary log. By contrast, row-based replication writes each changed row to the binary log. If the statement changes many rows, row-based replication may write significantly more data to the binary log; this is true even for statements that are rolled back. This also means that making and restoring a backup can require more time. In addition, the binary log is locked for a longer time to write the data, which may cause concurrency problems. Use [**binlog\_row\_image=minimal**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_image) to reduce the disadvantage considerably.

Deterministic UDFs that generate large [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) values take longer to replicate with row-based replication than with statement-based replication. This is because the [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column value is logged, rather than the statement generating the data.

You cannot see on the replica what statements were received from the source and executed. However, you can see what data was changed using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) with the options [--base64-output=DECODE-ROWS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_base64-output) and [--verbose](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_verbose).

Alternatively, use the [**binlog\_rows\_query\_log\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rows_query_log_events) variable, which if enabled adds a **Rows\_query** event with the statement to [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output when the **-vv** option is used.

For tables using the [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) storage engine, a stronger lock is required on the replica for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements when applying them as row-based events to the binary log than when applying them as statements. This means that concurrent inserts on [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables are not supported when using row-based replication.

**17.2.1.2 Usage of Row-Based Logging and Replication**

MySQL uses statement-based logging (SBL), row-based logging (RBL) or mixed-format logging. The type of binary log used impacts the size and efficiency of logging. Therefore the choice between row-based replication (RBR) or statement-based replication (SBR) depends on your application and environment. This section describes known issues when using a row-based format log, and describes some best practices using it in replication.

For additional information, see [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats), and [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

For information about issues specific to NDB Cluster Replication (which depends on row-based replication), see [Section 23.6.3, “Known Issues in NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-issues).

**Row-based logging of temporary tables.** As noted in [Section 17.5.1.31, “Replication and Temporary Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-temptables), temporary tables are not replicated when using row-based format or (from MySQL 8.0.4) mixed format. For more information, see [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

Temporary tables are not replicated when using row-based or mixed format because there is no need. In addition, because temporary tables can be read only from the thread which created them, there is seldom if ever any benefit obtained from replicating them, even when using statement-based format.

You can switch from statement-based to row-based binary logging format at runtime even when temporary tables have been created. However, in MySQL 8.0, you cannot switch from row-based or mixed format for binary logging to statement-based format at runtime, due to any **CREATE TEMPORARY TABLE** statements having been omitted from the binary log in the previous mode.

The MySQL server tracks the logging mode that was in effect when each temporary table was created. When a given client session ends, the server logs a **DROP TEMPORARY TABLE IF EXISTS** statement for each temporary table that still exists and was created when statement-based binary logging was in use. If row-based or mixed format binary logging was in use when the table was created, the **DROP TEMPORARY TABLE IF EXISTS** statement is not logged. In releases before MySQL 8.0.4 and 5.7.25, the **DROP TEMPORARY TABLE IF EXISTS** statement was logged regardless of the logging mode that was in effect.

Nontransactional DML statements involving temporary tables are allowed when using [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format), as long as any nontransactional tables affected by the statements are temporary tables (Bug #14272672).

**RBL and synchronization of nontransactional tables.** When many rows are affected, the set of changes is split into several events; when the statement commits, all of these events are written to the binary log. When executing on the replica, a table lock is taken on all tables involved, and then the rows are applied in batch mode. Depending on the engine used for the replica's copy of the table, this may or may not be effective.

**Latency and binary log size.** RBL writes changes for each row to the binary log and so its size can increase quite rapidly. This can significantly increase the time required to make changes on the replica that match those on the source. You should be aware of the potential for this delay in your applications.

**Reading the binary log.** [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) displays row-based events in the binary log using the **BINLOG** statement (see [Section 13.7.8.1, “BINLOG Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog)). This statement displays an event as a base 64-encoded string, the meaning of which is not evident. When invoked with the [--base64-output=DECODE-ROWS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_base64-output) and [--verbose](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_verbose) options, [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) formats the contents of the binary log to be human readable. When binary log events were written in row-based format and you want to read or recover from a replication or database failure you can use this command to read contents of the binary log. For more information, see [Section 4.6.9.2, “mysqlbinlog Row Event Display”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog-row-events).

**Binary log execution errors and replica execution mode.** Using [**slave\_exec\_mode=IDEMPOTENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) is generally only useful with MySQL NDB Cluster replication, for which **IDEMPOTENT** is the default value. (See [Section 23.6.10, “NDB Cluster Replication: Bidrectional and Circular Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-multi-source)). When [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) is **IDEMPOTENT**, a failure to apply changes from RBL because the original row cannot be found does not trigger an error or cause replication to fail. This means that it is possible that updates are not applied on the replica, so that the source and replica are no longer synchronized. Latency issues and use of nontransactional tables with RBR when [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) is **IDEMPOTENT** can cause the source and replica to diverge even further. For more information about [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode), see [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

For other scenarios, setting [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) to **STRICT** is normally sufficient; this is the default value for storage engines other than [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html).

**Filtering based on server ID not supported.** You can filter based on server ID by using the **IGNORE\_SERVER\_IDS** option for the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). This option works with statement-based and row-based logging formats, but is deprecated for use when [**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set. Another method to filter out changes on some replicas is to use a **WHERE** clause that includes the relation **@@server\_id <> *id\_value*** clause with [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements. For example, **WHERE @@server\_id <> 1**. However, this does not work correctly with row-based logging. To use the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable for statement filtering, use statement-based logging.

**RBL, nontransactional tables, and stopped replicas.** When using row-based logging, if the replica server is stopped while a replica thread is updating a nontransactional table, the replica database can reach an inconsistent state. For this reason, it is recommended that you use a transactional storage engine such as [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) for all tables replicated using the row-based format. Use of [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) or [**STOP REPLICA | SLAVE SQL\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) prior to shutting down the replica MySQL server helps prevent issues from occurring, and is always recommended regardless of the logging format or storage engine you use.

**17.2.1.3 Determination of Safe and Unsafe Statements in Binary Logging**

The “safeness” of a statement in MySQL replication refers to whether the statement and its effects can be replicated correctly using statement-based format. If this is true of the statement, we refer to the statement as safe; otherwise, we refer to it as unsafe.

In general, a statement is safe if it deterministic, and unsafe if it is not. However, certain nondeterministic functions are *not* considered unsafe (see [Nondeterministic functions not considered unsafe](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-safe-unsafe-not), later in this section). In addition, statements using results from floating-point math functions—which are hardware-dependent—are always considered unsafe (see [Section 17.5.1.12, “Replication and Floating-Point Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-floatvalues)).

**Handling of safe and unsafe statements.** A statement is treated differently depending on whether the statement is considered safe, and with respect to the binary logging format (that is, the current value of [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)).

When using row-based logging, no distinction is made in the treatment of safe and unsafe statements.

When using mixed-format logging, statements flagged as unsafe are logged using the row-based format; statements regarded as safe are logged using the statement-based format.

When using statement-based logging, statements flagged as being unsafe generate a warning to this effect. Safe statements are logged normally.

Each statement flagged as unsafe generates a warning. If a large number of such statements were executed on the source, this could lead to excessively large error log files. To prevent this, MySQL has a warning suppression mechanism. Whenever the 50 most recent [**ER\_BINLOG\_UNSAFE\_STATEMENT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_binlog_unsafe_statement) warnings have been generated more than 50 times in any 50-second period, warning suppression is enabled. When activated, this causes such warnings not to be written to the error log; instead, for each 50 warnings of this type, a note **The last warning was repeated *N* times in last *S* seconds** is written to the error log. This continues as long as the 50 most recent such warnings were issued in 50 seconds or less; once the rate has decreased below this threshold, the warnings are once again logged normally. Warning suppression has no effect on how the safety of statements for statement-based logging is determined, nor on how warnings are sent to the client. MySQL clients still receive one warning for each such statement.

For more information, see [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).

**Statements considered unsafe.** Statements with the following characteristics are considered unsafe:

**Statements containing system functions that may return a different value on the replica.** These functions include [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows), [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock), [**IS\_FREE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-free-lock), [**IS\_USED\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-used-lock), [**LOAD\_FILE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file), [**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait), [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand), [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock), [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count), [**SESSION\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_session-user), [**SLEEP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sleep), [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate), [**SYSTEM\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_system-user), [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user), [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid), and [**UUID\_SHORT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid-short).

**Nondeterministic functions not considered unsafe.** Although these functions are not deterministic, they are treated as safe for purposes of logging and replication: [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id), [**CURDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curdate), [**CURRENT\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-date), [**CURRENT\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-time), [**CURRENT\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp), [**CURTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_curtime),, [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), [**LOCALTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtime), [**LOCALTIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_localtimestamp), [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now), [**UNIX\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_unix-timestamp), [**UTC\_DATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-date), [**UTC\_TIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-time), and [**UTC\_TIMESTAMP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_utc-timestamp).

For more information, see [Section 17.5.1.14, “Replication and System Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-functions).

**References to system variables.** Most system variables are not replicated correctly using the statement-based format. See [Section 17.5.1.39, “Replication and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-variables). For exceptions, see [Section 5.4.4.3, “Mixed Binary Logging Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-mixed).

**UDFs.** Since we have no control over what a UDF does, we must assume that it is executing unsafe statements.

**Fulltext plugin.** This plugin may behave differently on different MySQL servers; therefore, statements depending on it could have different results. For this reason, all statements relying on the fulltext plugin are treated as unsafe in MySQL.

**Trigger or stored program updates a table having an AUTO\_INCREMENT column.** This is unsafe because the order in which the rows are updated may differ on the source and the replica.

In addition, an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) into a table that has a composite primary key containing an **AUTO\_INCREMENT** column that is not the first column of this composite key is unsafe.

For more information, see [Section 17.5.1.1, “Replication and AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-auto-increment).

**INSERT ... ON DUPLICATE KEY UPDATE statements on tables with multiple primary or unique keys.** When executed against a table that contains more than one primary or unique key, this statement is considered unsafe, being sensitive to the order in which the storage engine checks the keys, which is not deterministic, and on which the choice of rows updated by the MySQL Server depends.

An [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statement against a table having more than one unique or primary key is marked as unsafe for statement-based replication. (Bug #11765650, Bug #58637)

**Updates using LIMIT.** The order in which rows are retrieved is not specified, and is therefore considered unsafe. See [Section 17.5.1.18, “Replication and LIMIT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-limit).

**Accesses or references log tables.** The contents of the system log table may differ between source and replica.

**Nontransactional operations after transactional operations.** Within a transaction, allowing any nontransactional reads or writes to execute after any transactional reads or writes is considered unsafe.

For more information, see [Section 17.5.1.35, “Replication and Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transactions).

**Accesses or references self-logging tables.** All reads and writes to self-logging tables are considered unsafe. Within a transaction, any statement following a read or write to self-logging tables is also considered unsafe.

**LOAD DATA statements.** [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) is treated as unsafe and when [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) the statement is logged in row-based format. When [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) does not generate a warning, unlike other unsafe statements.

**XA transactions.** If two XA transactions committed in parallel on the source are being prepared on the replica in the inverse order, locking dependencies can occur with statement-based replication that cannot be safely resolved, and it is possible for replication to fail with deadlock on the replica. When [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, DML statements inside XA transactions are flagged as being unsafe and generate a warning. When [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) or [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, DML statements inside XA transactions are logged using row-based replication, and the potential issue is not present.

**DEFAULT clause that refers to a nondeterministic function.** If an expression default value refers to a nondeterministic function, any statement that causes the expression to be evaluated is unsafe for statement-based replication. This includes statements such as [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Unlike most other unsafe statements, this category of statement cannot be replicated safely in row-based format. When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**, the statement is logged and executed but a warning message is written to the error log. When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **MIXED** or **ROW**, the statement is not executed and an error message is written to the error log. For more information on the handling of explicit defaults, see [Explicit Default Handling as of MySQL 8.0.13](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults-explicit).

For additional information, see [Section 17.5.1, “Replication Features and Issues”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features).

**17.2.2 Replication Channels**

[17.2.2.1 Commands for Operations on a Single Channel](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-commands-single-channel)

[17.2.2.2 Compatibility with Previous Replication Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-with-prev-replication)

[17.2.2.3 Startup Options and Replication Channels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-startup-options)

[17.2.2.4 Replication Channel Naming Conventions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-naming-conventions)

In MySQL multi-source replication, a replica opens multiple replication channels, one for each source server. The replication channels represent the path of transactions flowing from a source to the replica. Each replication channel has its own receiver (I/O) thread, one or more applier (SQL) threads, and relay log. When transactions from a source are received by a channel's receiver thread, they are added to the channel's relay log file and passed through to the channel's applier threads. This enables each channel to function independently.

This section describes how channels can be used in a replication topology, and the impact they have on single-source replication. For instructions to configure sources and replicas for multi-source replication, to start, stop and reset multi-source replicas, and to monitor multi-source replication, see [Section 17.1.5, “MySQL Multi-Source Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source).

The maximum number of channels that can be created on one replica server in a multi-source replication topology is 256. Each replication channel must have a unique (nonempty) name, as explained in [Section 17.2.2.4, “Replication Channel Naming Conventions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-naming-conventions). The error codes and messages that are issued when multi-source replication is enabled specify the channel that generated the error.

**Note**

Each channel on a multi-source replica must replicate from a different source. You cannot set up multiple replication channels from a single replica to a single source. This is because the server IDs of replicas must be unique in a replication topology. The source distinguishes replicas only by their server IDs, not by the names of the replication channels, so it cannot recognize different replication channels from the same replica.

A multi-source replica can also be set up as a multi-threaded replica, by setting the [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) system variable to a value greater than 0. When you do this on a multi-source replica, each channel on the replica has the specified number of applier threads, plus a coordinator thread to manage them. You cannot configure the number of applier threads for individual channels.

From MySQL 8.0, multi-source replicas can be configured with replication filters on specific replication channels. Channel specific replication filters can be used when the same database or table is present on multiple sources, and you only need the replica to replicate it from one source. For GTID-based replication, if the same transaction might arrive from multiple sources (such as in a diamond topology), you must ensure the filtering setup is the same on all channels. For more information, see [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters).

To provide compatibility with previous versions, the MySQL server automatically creates on startup a default channel whose name is the empty string (**""**). This channel is always present; it cannot be created or destroyed by the user. If no other channels (having nonempty names) have been created, replication statements act on the default channel only, so that all replication statements from older replicas function as expected (see [Section 17.2.2.2, “Compatibility with Previous Replication Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-with-prev-replication). Statements applying to replication channels as described in this section can be used only when there is at least one named channel.

**17.2.2.1 Commands for Operations on a Single Channel**

To enable MySQL replication operations to act on individual replication channels, use the **FOR CHANNEL *channel*** clause with the following replication statements:

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to)

[**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to)

[**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica)

[**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events)

[**FLUSH RELAY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush)

[**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status)

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica)

An additional **channel** parameter is introduced for the following function:

[**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait)

The following statements are disallowed for the **group\_replication\_recovery** channel:

[**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica)

The following statements are disallowed for the **group\_replication\_applier** channel:

[**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica)

[**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status)

[**FLUSH RELAY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) is now permitted for the **group\_replication\_applier** channel, but if the request is received while a transaction is being applied, the request is performed after the transaction ends. The requester must wait while the transaction is completed and the rotation takes place. This behavior prevents transactions from being split, which is not permitted for Group Replication.

**17.2.2.2 Compatibility with Previous Replication Statements**

When a replica has multiple channels and a **FOR CHANNEL *channel*** option is not specified, a valid statement generally acts on all available channels, with some specific exceptions.

For example, the following statements behave as expected for all except certain Group Replication channels:

[**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) starts replication threads for all channels, except the **group\_replication\_recovery** and **group\_replication\_applier** channels.

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) stops replication threads for all channels, except the **group\_replication\_recovery** and **group\_replication\_applier** channels.

[**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) reports the status for all channels, except the **group\_replication\_applier** channel.

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets all channels.

**Warning**

Use [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) with caution as this statement deletes all existing channels, purges their relay log files, and recreates only the default channel.

Some replication statements cannot operate on all channels. In this case, error 1964 Multiple channels exist on the replica. Please provide channel name as an argument. is generated. The following statements and functions generate this error when used in a multi-source replication topology and a **FOR CHANNEL *channel*** option is not used to specify which channel to act on:

[**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events)

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to)

[**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to)

[**MASTER\_POS\_WAIT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_master-pos-wait)

Note that a default channel always exists in a single source replication topology, where statements and functions behave as in previous versions of MySQL.

**17.2.2.3 Startup Options and Replication Channels**

This section describes startup options which are impacted by the addition of replication channels.

The [**master\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) and [**relay\_log\_info\_repository**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) system variables must *not* be set to **FILE** when you use replication channels. In MySQL 8.0, the **FILE** setting is deprecated, and **TABLE** is the default, so the system variables can be omitted. From MySQL 8.0.23, they must be omitted because their use is deprecated from that release. If these system variables are set to **FILE**, attempting to add more sources to a replica fails with [**ER\_SLAVE\_NEW\_CHANNEL\_WRONG\_REPOSITORY**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_slave_new_channel_wrong_repository).

The following startup options now affect *all* channels in a replication topology.

[**--log-slave-updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates)

All transactions received by the replica (even from multiple sources) are written in the binary log.

[**--relay-log-purge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_purge)

When set, each channel purges its own relay log automatically.

[**--slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries)

The specified number of transaction retries can take place on all applier threads of all channels.

[--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) (or [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable set)

No replication threads start on any channels.

[**--slave-skip-errors**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_skip_errors)

Execution continues and errors are skipped for all channels.

The values set for the following startup options apply on each channel; since these are [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) startup options, they are applied on every channel.

--max-relay-log-size=***size***

Maximum size of the individual relay log file for each channel; after reaching this limit, the file is rotated.

--relay-log-space-limit=***size***

Upper limit for the total size of all relay logs combined, for each individual channel. For ***N*** channels, the combined size of these logs is limited to [**relay\_log\_space\_limit \* *N***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_space_limit).

--slave-parallel-workers=***value***

Number of replication applier threads per channel.

[**slave\_checkpoint\_group**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_checkpoint_group)

Waiting time by an I/O thread for each source.

--relay-log-index=filename

Base name for each channel's relay log index file. See [Section 17.2.2.4, “Replication Channel Naming Conventions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-naming-conventions).

--relay-log=filename

Denotes the base name of each channel's relay log file. See [Section 17.2.2.4, “Replication Channel Naming Conventions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#channels-naming-conventions).

--slave\_net-timeout=N

This value is set per channel, so that each channel waits for ***N*** seconds to check for a broken connection.

--slave-skip-counter=N

This value is set per channel, so that each channel skips ***N*** events from its source.

**17.2.2.4 Replication Channel Naming Conventions**

This section describes how naming conventions are impacted by replication channels.

Each replication channel has a unique name which is a string with a maximum length of 64 characters and is case-insensitive. Because channel names are used in the replica's applier metadata repository table, the character set used for these is always UTF-8. Although you are generally free to use any name for channels, the following names are reserved:

**group\_replication\_applier**

**group\_replication\_recovery**

The name you choose for a replication channel also influences the file names used by a multi-source replica. The relay log files and index files for each channel are named ***relay\_log\_basename***-***channel***.xxxxxx, where ***relay\_log\_basename*** is a base name specified using the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable, and ***channel*** is the name of the channel logged to this file. If you do not specify the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable, a default file name is used that also includes the name of the channel.

**17.2.3 Replication Threads**

[17.2.3.1 Monitoring Replication Main Threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-threads-monitor-main)

[17.2.3.2 Monitoring Replication Applier Worker Threads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-threads-monitor-worker)

MySQL replication capabilities are implemented using three main threads, one on the source server and two on the replica:

**Binary log dump thread.** The source creates a thread to send the binary log contents to a replica when the replica connects. This thread can be identified in the output of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) on the source as the **Binlog Dump** thread.

The binary log dump thread acquires a lock on the source's binary log for reading each event that is to be sent to the replica. As soon as the event has been read, the lock is released, even before the event is sent to the replica.

**Replication I/O thread.** When a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement is issued on a replica server, the replica creates an I/O thread, which connects to the source and asks it to send the updates recorded in its binary logs.

The replication I/O thread reads the updates that the source's **Binlog Dump** thread sends (see previous item) and copies them to local files that comprise the replica's relay log.

The state of this thread is shown as **Slave\_IO\_running** in the output of [**SHOW SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status).

**Replication SQL thread.** The replica creates an SQL thread to read the relay log that is written by the replication I/O thread and execute the transactions contained in it.

There are three main threads for each source/replica connection. A source that has multiple replicas creates one binary log dump thread for each currently connected replica, and each replica has its own replication I/O and SQL threads.

A replica uses two threads to separate reading updates from the source and executing them into independent tasks. Thus, the task of reading transactions is not slowed down if the process of applying them is slow. For example, if the replica server has not been running for a while, its I/O thread can quickly fetch all the binary log contents from the source when the replica starts, even if the SQL thread lags far behind. If the replica stops before the SQL thread has executed all the fetched statements, the I/O thread has at least fetched everything so that a safe copy of the transactions is stored locally in the replica's relay logs, ready for execution the next time that the replica starts.

You can enable further parallelization for tasks on a replica by setting the [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) system variable to a value greater than 0 (the default). When this system variable is set, the replica creates the specified number of worker threads to apply transactions, plus a coordinator thread to manage them. If you are using multiple replication channels, each channel has this number of threads. A replica with [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) set to a value greater than 0 is called a multithreaded replica. With this setup, transactions that fail can be retried.

**Note**

Multithreaded replicas are not currently supported by NDB Cluster, which silently ignores the setting for this variable. See [Section 23.6.3, “Known Issues in NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication-issues) for more information.

**17.2.3.1 Monitoring Replication Main Threads**

The [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) statement provides information that tells you what is happening on the source and on the replica regarding replication. For information on source states, see [Section 8.14.4, “Replication Source Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#source-thread-states). For replica states, see [Section 8.14.5, “Replication I/O Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-io-thread-states), and [Section 8.14.6, “Replication SQL Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-sql-thread-states).

The following example illustrates how the three main replication threads, the binary log dump thread, replicatin I/O thread, and replication SQL thread, show up in the output from [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist).

On the source server, the output from [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) looks like this:

mysql> **SHOW PROCESSLIST\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 2

User: root

Host: localhost:32931

db: NULL

Command: Binlog Dump

Time: 94

State: Has sent all binlog to slave; waiting for binlog to

be updated

Info: NULL

Here, thread 2 is a **Binlog Dump** thread that services a connected replica. The **State** information indicates that all outstanding updates have been sent to the replica and that the source is waiting for more updates to occur. If you see no **Binlog Dump** threads on a source server, this means that replication is not running; that is, no replicas are currently connected.

On a replica server, the output from [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) looks like this:

mysql> **SHOW PROCESSLIST\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 10

User: system user

Host:

db: NULL

Command: Connect

Time: 11

State: Waiting for master to send event

Info: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 11

User: system user

Host:

db: NULL

Command: Connect

Time: 11

State: Has read all relay log; waiting for the slave I/O

thread to update it

Info: NULL

The **State** information indicates that thread 10 is the replication I/O thread that is communicating with the source server, and thread 11 is the replication SQL thread that is processing the updates stored in the relay logs. At the time that [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) was run, both threads were idle, waiting for further updates.

The value in the **Time** column can show how late the replica is compared to the source. See [Section A.14, “MySQL 8.0 FAQ: Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\faqs.html#faqs-replication). If sufficient time elapses on the source side without activity on the **Binlog Dump** thread, the source determines that the replica is no longer connected. As for any other client connection, the timeouts for this depend on the values of net\_write\_timeout and net\_retry\_count; for more information about these, see [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

The [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement provides additional information about replication processing on a replica server. See [Section 17.1.7.1, “Checking Replication Status”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-status).

**17.2.3.2 Monitoring Replication Applier Worker Threads**

On a multithreaded replica, the Performance Schema tables [**replication\_applier\_status\_by\_coordinator**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-coordinator-table) and [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) show status information for the replica's coordinator thread and applier worker threads respectively. For a replica with multiple channels, the threads for each channel are identified.

A multithreaded replica's coordinator thread also prints statistics to the replica's error log on a regular basis if the verbosity setting is set to display informational messages. The statistics are printed depending on the volume of events that the coordinator thread has assigned to applier worker threads, with a maximum frequency of once every 120 seconds. The message lists the following statistics for the relevant replication channel, or the default replication channel (which is not named):

**Seconds elapsed**

The difference in seconds between the current time and the last time this information was printed to the error log.

**Events assigned**

The total number of events that the coordinator thread has queued to all applier worker threads since the coordinator thread was started.

**Worker queues filled over overrun level**

The current number of events that are queued to any of the applier worker threads in excess of the overrun level, which is set at 90% of the maximum queue length of 16384 events. If this value is zero, no applier worker threads are operating at the upper limit of their capacity.

**Waited due to worker queue full**

The number of times that the coordinator thread had to wait to schedule an event because an applier worker thread's queue was full. If this value is zero, no applier worker threads exhausted their capacity.

**Waited due to the total size**

The number of times that the coordinator thread had to wait to schedule an event because the [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) limit had been reached. This system variable sets the maximum amount of memory (in bytes) available to applier worker thread queues holding events not yet applied. If an unusually large event exceeds this size, the transaction is held until all the applier worker threads have empty queues, and then processed. All subsequent transactions are held until the large transaction has been completed.

**Waited at clock conflicts**

The number of nanoseconds that the coordinator thread had to wait to schedule an event because a transaction that the event depended on had not yet been committed. If [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **DATABASE** (rather than **LOGICAL\_CLOCK**), this value is always zero.

**Waited (count) when workers occupied**

The number of times that the coordinator thread slept for a short period, which it might do in two situations. The first situation is where the coordinator thread assigns an event and finds the applier worker thread's queue is filled beyond the underrun level of 10% of the maximum queue length, in which case it sleeps for a maximum of 1 millisecond. The second situation is where [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **LOGICAL\_CLOCK** and the coordinator thread needs to assign the first event of a transaction to an applier worker thread's queue, it only does this to a worker with an empty queue, so if no queues are empty, the coordinator thread sleeps until one becomes empty.

**Waited when workers occupied**

The number of nanoseconds that the coordinator thread slept while waiting for an empty applier worker thread queue (that is, in the second situation described above, where [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **LOGICAL\_CLOCK** and the first event of a transaction needs to be assigned).

**17.2.4 Relay Log and Replication Metadata Repositories**

[17.2.4.1 The Relay Log](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-relaylog)

[17.2.4.2 Replication Metadata Repositories](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status)

A replica server creates several repositories of information to use for the replication process:

The replica's *relay log*, which is written by the replication I/O thread, contains the transactions read from the replication source server's binary log. The transactions in the relay log are applied on the replica by the replication SQL thread. For information about the relay log, see [Section 17.2.4.1, “The Relay Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-relaylog).

The replica's *connection metadata repository* contains information that the replication I/O thread needs to connect to the replication source server and retrieve transactions from the source's binary log. The connection metadata repository is written to the **mysql.slave\_master\_info** table.

The replica's *applier metadata repository* contains information that the replication SQL thread needs to read and apply transactions from the replica's relay log. The applier metadata repository is written to the **mysql.slave\_relay\_log\_info** table.

The replica's connection metadata repository and applier metadata repository are collectively known as the replication metadata repositories. For information about these, see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status).

**Making replication resilient to unexpected halts.** The **mysql.slave\_master\_info** and **mysql.slave\_relay\_log\_info** tables are created using the transactional storage engine [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html). Updates to the replica's applier metadata repository table are committed together with the transactions, meaning that the replica's progress information recorded in that repository is always consistent with what has been applied to the database, even in the event of an unexpected server halt. For information on the combination of settings on the replica that is most resilient to unexpected halts, see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

**17.2.4.1 The Relay Log**

The relay log, like the binary log, consists of a set of numbered files containing events that describe database changes, and an index file that contains the names of all used relay log files. The default location for relay log files is the data directory.

The term “relay log file” generally denotes an individual numbered file containing database events. The term “relay log” collectively denotes the set of numbered relay log files plus the index file.

Relay log files have the same format as binary log files and can be read using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) (see [Section 4.6.9, “mysqlbinlog — Utility for Processing Binary Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog)). If binary log transaction compression (available as of MySQL 8.0.20) is in use, transaction payloads written to the relay log are compressed in the same way as for the binary log. For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

For the default replication channel, relay log file names have the default form ***host\_name***-relay-bin.***nnnnnn***, where ***host\_name*** is the name of the replica server host and ***nnnnnn*** is a sequence number. Successive relay log files are created using successive sequence numbers, beginning with **000001**. For non-default replication channels, the default base name is ***host\_name***-relay-bin-***channel***, where ***channel*** is the name of the replication channel recorded in the relay log.

The replica uses an index file to track the relay log files currently in use. The default relay log index file name is ***host\_name***-relay-bin.index for the default channel, and ***host\_name***-relay-bin-***channel***.index for non-default replication channels.

The default relay log file and relay log index file names and locations can be overridden with, respectively, the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) and [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variables (see [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options)).

If a replica uses the default host-based relay log file names, changing a replica's host name after replication has been set up can cause replication to fail with the errors Failed to open the relay log and Could not find target log during relay log initialization. This is a known issue (see Bug #2122). If you anticipate that a replica's host name might change in the future (for example, if networking is set up on the replica such that its host name can be modified using DHCP), you can avoid this issue entirely by using the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) and [**relay\_log\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_index) system variables to specify relay log file names explicitly when you initially set up the replica. This causes the names to be independent of server host name changes.

If you encounter the issue after replication has already begun, one way to work around it is to stop the replica server, prepend the contents of the old relay log index file to the new one, and then restart the replica. On a Unix system, this can be done as shown here:

shell> **cat *new\_relay\_log\_name*.index >> *old\_relay\_log\_name*.index**

shell> **mv *old\_relay\_log\_name*.index *new\_relay\_log\_name*.index**

A replica server creates a new relay log file under the following conditions:

Each time the replication I/O thread starts.

When the logs are flushed (for example, with [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) or [**mysqladmin flush-logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin)).

When the size of the current relay log file becomes too large, which is determined as follows:

If the value of [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) is greater than 0, that is the maximum relay log file size.

If the value of [**max\_relay\_log\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_relay_log_size) is 0, [**max\_binlog\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_max_binlog_size) determines the maximum relay log file size.

The replication SQL thread automatically deletes each relay log file after it has executed all events in the file and no longer needs it. There is no explicit mechanism for deleting relay logs because the replication SQL thread takes care of doing so. However, [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) rotates relay logs, which influences when the replication SQL thread deletes them.

**17.2.4.2 Replication Metadata Repositories**

A replica server creates two replication metadata repositories, the connection metadata repository and the applier metadata repository. The replication metadata repositories survive a replica server's shutdown. If binary log file position based replication is in use, when the replica restarts, it reads the two repositories to determine how far it previously proceeded in reading the binary log from the source and in processing its own relay log. If GTID-based replication is in use, the replica does not use the replication metadata repositories for that purpose, but does need them for the other metadata that they contain.

The replica's *connection metadata repository* contains information that the replication I/O thread needs to connect to the replication source server and retrieve transactions from the source's binary log. The metadata in this repository includes the connection configuration, the replication user account details, the SSL settings for the connection, and the file name and position where the replication I/O thread is currently reading from the source's binary log.

The replica's *applier metadata repository* contains information that the replication SQL thread needs to read and apply transactions from the replica's relay log. The metadata in this repository includes the file name and position up to which the replication SQL thread has executed the transactions in the relay log, and the equivalent position in the source's binary log. It also includes metadata for the process of applying transactions, such as the number of worker threads and the **PRIVILEGE\_CHECKS\_USER** account for the channel.

The connection metadata repository is written to the **slave\_master\_info** table in the **mysql** system schema, and the applier metadata repository is written to the **slave\_relay\_log\_info** table in the **mysql** system schema. A warning message is issued if [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is unable to initialize the tables for the replication metadata repositories, but the replica is allowed to continue starting. This situation is most likely to occur when upgrading from a version of MySQL that does not support the use of tables for the repositories to one in which they are supported.

**Important**

Do not attempt to update or insert rows in the **mysql.slave\_master\_info** or **mysql.slave\_relay\_log\_info** tables manually. Doing so can cause undefined behavior, and is not supported. Execution of any statement requiring a write lock on either or both of the **slave\_master\_info** and **slave\_relay\_log\_info** tables is disallowed while replication is ongoing (although statements that perform only reads are permitted at any time).

Access privileges for the connection metadata repository table **mysql.slave\_master\_info** should be restricted to the database administrator, because it contains the replication user account name and password for connecting to the source. Use a restricted access mode to protect database backups that include this table. From MySQL 8.0.21, you can clear the replication user account credentials from the connection metadata repository, and instead always provide them using the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement or [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement that starts the replication channel. This approach means that the replication channel always needs operator intervention to restart, but the account name and password are not recorded in the replication metadata repositories.

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) clears the data in the replication metadata repositories, with the exception of the replication connection parameters (depending on the MySQL Server release). For details, see the description for [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica).

Before MySQL 8.0, to create the replication metadata repositories as tables, it was necessary to specify [**master\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) and [**relay\_log\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) at server startup. Otherwise, the repositories were created as files in the data directory named master.info and relay-log.info, or with alternative names and locations specified by the [--master-info-file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-info-file) option and [**relay\_log\_info\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_file) system variable. From MySQL 8.0, creating the replication metadata repositories as tables is the default, and the use of all these system variables is deprecated.

The **mysql.slave\_master\_info** and **mysql.slave\_relay\_log\_info** tables are created using the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) transactional storage engine. Updates to the applier metadata repository table are committed together with the transactions, meaning that the replica's progress information recorded in that repository is always consistent with what has been applied to the database, even in the event of an unexpected server halt. For information on the combination of settings on a replica that is most resilient to unexpected halts, see [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

When you back up the replica's data or transfer a snapshot of its data to create a new replica, ensure that you include the **mysql.slave\_master\_info** and **mysql.slave\_relay\_log\_info** tables containing the replication metadata repositories. For cloning operations, note that when the replication metadata repositories are created as tables, they are copied to the recipient during a cloning operation, but when they are created as files, they are not copied. When binary log file position based replication is in use, the replication metadata repositories are needed to resume replication after restarting the restored, copied, or cloned replica. If you do not have the relay log files, but still have the applier metadata repository, you can check it to determine how far the replication SQL thread has executed in the source's binary log. Then you can use a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) with the **SOURCE\_LOG\_FILE** | **MASTER\_LOG\_FILE** and **SOURCE\_LOG\_POS** | **MASTER\_LOG\_POS** options to tell the replica to re-read the binary logs from the source from that point (provided that the required binary logs still exist on the source).

One additional repository, the applier worker metadata repository, is created primarily for internal use, and holds status information about worker threads on a multithreaded replica. The applier worker metadata repository includes the names and positions for the relay log file and the source's binary log file for each worker thread. If the applier metadata repository is created as a table, which is the default, the applier worker metadata repository is written to the **mysql.slave\_worker\_info** table. If the applier metadata repository is written to a file, the applier worker metadata repository is written to the worker-relay-log.info file. For external use, status information for worker threads is presented in the Performance Schema [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) table.

The replication metadata repositories originally contained information similar to that shown in the output of the [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement, which is discussed in [Section 13.4.2, “SQL Statements for Controlling Replica Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-replica). Further information has since been added to the replication metadata repositories which is not displayed by the [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement.

For the connection metadata repository, the following table shows the correspondence between the columns in the **mysql.slave\_master\_info** table, the columns displayed by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status), and the lines in the deprecated master.info file.

| **slave\_master\_info Table Column** | **SHOW REPLICA | SLAVE STATUS Column** | master.info**File Line** | **Description** |
| --- | --- | --- | --- |
| **Number\_of\_lines** | [None] | 1 | Number of columns in the table (or lines in the file) |
| **Master\_log\_name** | **Source\_Log\_File** | 2 | The name of the binary log currently being read from the source |
| **Master\_log\_pos** | **Read\_Source\_Log\_Pos** | 3 | The current position within the binary log that has been read from the source |
| **Host** | **Source\_Host** | 4 | The host name of the replication source server |
| **User\_name** | **Source\_User** | 5 | The replication user account name used to connect to the source |
| **User\_password** | Password (not shown by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status)) | 6 | The replication user account password used to connect to the source |
| **Port** | **Source\_Port** | 7 | The network port used to connect to the replication source server |
| **Connect\_retry** | **Connect\_Retry** | 8 | The period (in seconds) that the replica waits before trying to reconnect to the source |
| **Enabled\_ssl** | **Source\_SSL\_Allowed** | 9 | Whether the replica supports SSL connections |
| **Ssl\_ca** | **Source\_SSL\_CA\_File** | 10 | The file used for the Certificate Authority (CA) certificate |
| **Ssl\_capath** | **Source\_SSL\_CA\_Path** | 11 | The path to the Certificate Authority (CA) certificate |
| **Ssl\_cert** | **Source\_SSL\_Cert** | 12 | The name of the SSL certificate file |
| **Ssl\_cipher** | **Source\_SSL\_Cipher** | 13 | The list of possible ciphers used in the handshake for the SSL connection |
| **Ssl\_key** | **Source\_SSL\_Key** | 14 | The name of the SSL key file |
| **Ssl\_verify\_server\_cert** | **Source\_SSL\_Verify\_Server\_Cert** | 15 | Whether to verify the server certificate |
| **Heartbeat** | [None] | 16 | Interval between replication heartbeats, in seconds |
| **Bind** | **Source\_Bind** | 17 | Which of the replica's network interfaces should be used for connecting to the source |
| **Ignored\_server\_ids** | **Replicate\_Ignore\_Server\_Ids** | 18 | The list of server IDs to be ignored. Note that for **Ignored\_server\_ids** the list of server IDs is preceded by the total number of server IDs to ignore. |
| **Uuid** | **Source\_UUID** | 19 | The source's unique ID |
| **Retry\_count** | **Source\_Retry\_Count** | 20 | Maximum number of reconnection attempts permitted |
| **Ssl\_crl** | [None] | 21 | Path to an SSL certificate revocation-list file |
| **Ssl\_crlpath** | [None] | 22 | Path to a directory containing SSL certificate revocation-list files |
| **Enabled\_auto\_position** | **Auto\_position** | 23 | Whether GTID auto-positioning is in use or not |
| **Channel\_name** | **Channel\_name** | 24 | The name of the replication channel |
| **Tls\_version** | **Source\_TLS\_Version** | 25 | TLS version on the source |
| **Public\_key\_path** | **Source\_public\_key\_path** | 26 | Name of the RSA public key file |
| **Get\_public\_key** | **Get\_source\_public\_key** | 27 | Whether to request RSA public key from source |
| **Network\_namespace** | **Network\_namespace** | 28 | Network namespace |
| **Master\_compression\_algorithm** | [None] | 29 | Permitted compression algorithms for the connection to the source |
| **Master\_zstd\_compression\_level** | [None] | 30 | **zstd** compression level |
| **Tls\_ciphersuites** | [None] | 31 | Permitted ciphersuites for TLSv1.3 |
| **Source\_connection\_auto\_failover** | [None] | 32 | Whether the asynchronous connection failover mechanism is activated |

For the applier metadata repository, the following table shows the correspondence between the columns in the **mysql.slave\_relay\_log\_info** table, the columns displayed by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status), and the lines in the deprecated relay-log.info file.

| **slave\_relay\_log\_info Table Column** | **SHOW REPLICA | SLAVE STATUS Column** | **Line in**relay-log.info**File** | **Description** |
| --- | --- | --- | --- |
| **Number\_of\_lines** | [None] | 1 | Number of columns in the table or lines in the file |
| **Relay\_log\_name** | **Relay\_Log\_File** | 2 | The name of the current relay log file |
| **Relay\_log\_pos** | **Relay\_Log\_Pos** | 3 | The current position within the relay log file; events up to this position have been executed on the replica database |
| **Master\_log\_name** | **Relay\_Source\_Log\_File** | 4 | The name of the source's binary log file from which the events in the relay log file were read |
| **Master\_log\_pos** | **Exec\_Source\_Log\_Pos** | 5 | The equivalent position within the source's binary log file of the events that have been executed on the replica |
| **Sql\_delay** | **SQL\_Delay** | 6 | The number of seconds that the replica must lag the source |
| **Number\_of\_workers** | [None] | 7 | The number of worker threads for applying replication transactions in parallel |
| **Id** | [None] | 8 | ID used for internal purposes; currently this is always 1 |
| **Channel\_name** | **Channel\_name** | 9 | The name of the replication channel |
| **Privilege\_checks\_username** | [None] | 10 | The user name for the **PRIVILEGE\_CHECKS\_USER** account for the channel |
| **Privilege\_checks\_hostname** | [None] | 11 | The host name for the **PRIVILEGE\_CHECKS\_USER** account for the channel |
| **Require\_row\_format** | [None] | 12 | Whether the channel accepts only row-based events |
| **Require\_table\_primary\_key\_check** | [None] | 13 | The channel's policy on whether tables must have primary keys for **CREATE TABLE** and **ALTER TABLE** operations |
| **Assign\_gtids\_to\_anonymous\_transactions\_type** | [None] | 14 | Whether the channel assigns a GTID to replicated transactions that do not already have one, and if so, whether it uses the replica's local UUID or a manually set UUID |
| **Assign\_gtids\_to\_anonymous\_transactions\_value** | [None] | 15 | The UUID used in the GTIDs assigned to anonymous transactions |

**17.2.5 How Servers Evaluate Replication Filtering Rules**

[17.2.5.1 Evaluation of Database-Level Replication and Binary Logging Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-db-options)

[17.2.5.2 Evaluation of Table-Level Replication Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-table-options)

[17.2.5.3 Interactions Between Replication Filtering Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-examples)

[17.2.5.4 Replication Channel Based Filters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters)

If a replication source server does not write a statement to its binary log, the statement is not replicated. If the server does log the statement, the statement is sent to all replicas and each replica determines whether to execute it or ignore it.

On the source, you can control which databases to log changes for by using the [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) and [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) options to control binary logging. For a description of the rules that servers use in evaluating these options, see [Section 17.2.5.1, “Evaluation of Database-Level Replication and Binary Logging Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-db-options). You should not use these options to control which databases and tables are replicated. Instead, use filtering on the replica to control the events that are executed on the replica.

On the replica side, decisions about whether to execute or ignore statements received from the source are made according to the --replicate-\* options that the replica was started with. (See [Section 17.1.6, “Replication and Binary Logging Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options).) The filters governed by these options can also be set dynamically using the **CHANGE REPLICATION FILTER** statement. The rules governing such filters are the same whether they are created on startup using --replicate-\* options or while the replica server is running by **CHANGE REPLICATION FILTER**. Note that replication filters cannot be used on Group Replication-specific channels on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state.

In the simplest case, when there are no --replicate-\* options, the replica executes all statements that it receives from the source. Otherwise, the result depends on the particular options given.

Database-level options ([--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db), [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db)) are checked first; see [Section 17.2.5.1, “Evaluation of Database-Level Replication and Binary Logging Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-db-options), for a description of this process. If no database-level options are used, option checking proceeds to any table-level options that may be in use (see [Section 17.2.5.2, “Evaluation of Table-Level Replication Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-table-options), for a discussion of these). If one or more database-level options are used but none are matched, the statement is not replicated.

For statements affecting databases only (that is, [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database), [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), and [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database)), database-level options always take precedence over any [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options. In other words, for such statements, [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options are checked if and only if there are no database-level options that apply.

To make it easier to determine what effect a given set of options has, it is recommended that you avoid mixing do-\* and ignore-\* options, or options containing wildcards with options which do not.

If any [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db) options were specified, they are applied before the --replicate-\* filtering rules are tested.

**Note**

All replication filtering options follow the same rules for case sensitivity that apply to names of databases and tables elsewhere in the MySQL server, including the effects of the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable.

**17.2.5.1 Evaluation of Database-Level Replication and Binary Logging Options**

When evaluating replication options, the replica begins by checking to see whether there are any [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) or [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) options that apply. When using [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) or [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db), the process is similar, but the options are checked on the source.

The database that is checked for a match depends on the binary log format of the statement that is being handled. If the statement has been logged using the row format, the database where data is to be changed is the database that is checked. If the statement has been logged using the statement format, the default database (specified with a [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement) is the database that is checked.

**Note**

Only DML statements can be logged using the row format. DDL statements are always logged as statements, even when [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format). All DDL statements are therefore always filtered according to the rules for statement-based replication. This means that you must select the default database explicitly with a [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement in order for a DDL statement to be applied.

For replication, the steps involved are listed here:

Which logging format is used?

**STATEMENT.** Test the default database.

**ROW.** Test the database affected by the changes.

Are there any [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) options?

**Yes.** Does the database match any of them?

**Yes.** Continue to Step 4.

**No.** Ignore the update and exit.

**No.** Continue to step 3.

Are there any [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) options?

**Yes.** Does the database match any of them?

**Yes.** Ignore the update and exit.

**No.** Continue to step 4.

**No.** Continue to step 4.

Proceed to checking the table-level replication options, if there are any. For a description of how these options are checked, see [Section 17.2.5.2, “Evaluation of Table-Level Replication Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-table-options).

**Important**

A statement that is still permitted at this stage is not yet actually executed. The statement is not executed until all table-level options (if any) have also been checked, and the outcome of that process permits execution of the statement.

For binary logging, the steps involved are listed here:

Are there any [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) or [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) options?

**Yes.** Continue to step 2.

**No.** Log the statement and exit.

Is there a default database (has any database been selected by [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use))?

**Yes.** Continue to step 3.

**No.** Ignore the statement and exit.

There is a default database. Are there any [--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) options?

**Yes.** Do any of them match the database?

**Yes.** Log the statement and exit.

**No.** Ignore the statement and exit.

**No.** Continue to step 4.

Do any of the [--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db) options match the database?

**Yes.** Ignore the statement and exit.

**No.** Log the statement and exit.

**Important**

For statement-based logging, an exception is made in the rules just given for the [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database), [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database), and [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) statements. In those cases, the database being *created, altered, or dropped* replaces the default database when determining whether to log or ignore updates.

[--binlog-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) can sometimes mean “ignore other databases”. For example, when using statement-based logging, a server running with only [--binlog-do-db=sales](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-do-db) does not write to the binary log statements for which the default database differs from **sales**. When using row-based logging with the same option, the server logs only those updates that change data in **sales**.

**17.2.5.2 Evaluation of Table-Level Replication Options**

The replica checks for and evaluates table options only if either of the following two conditions is true:

No matching database options were found.

One or more database options were found, and were evaluated to arrive at an “execute” condition according to the rules described in the previous section (see [Section 17.2.5.1, “Evaluation of Database-Level Replication and Binary Logging Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-db-options)).

First, as a preliminary condition, the replica checks whether statement-based replication is enabled. If so, and the statement occurs within a stored function, the replica executes the statement and exits. If row-based replication is enabled, the replica does not know whether a statement occurred within a stored function on the source, so this condition does not apply.

**Note**

For statement-based replication, replication events represent statements (all changes making up a given event are associated with a single SQL statement); for row-based replication, each event represents a change in a single table row (thus a single statement such as **UPDATE mytable SET mycol = 1** may yield many row-based events). When viewed in terms of events, the process of checking table options is the same for both row-based and statement-based replication.

Having reached this point, if there are no table options, the replica simply executes all events. If there are any [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) or [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options, the event must match one of these if it is to be executed; otherwise, it is ignored. If there are any [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table) or [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) options, all events are executed except those that match any of these options.

The following steps describe this evaluation in more detail. The starting point is the end of the evaluation of the database-level options, as described in [Section 17.2.5.1, “Evaluation of Database-Level Replication and Binary Logging Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-db-options).

Are there any table replication options?

**Yes.** Continue to step 2.

**No.** Execute the update and exit.

Which logging format is used?

**STATEMENT.** Carry out the remaining steps for each statement that performs an update.

**ROW.** Carry out the remaining steps for each update of a table row.

Are there any [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) options?

**Yes.** Does the table match any of them?

**Yes.** Execute the update and exit.

**No.** Continue to step 4.

**No.** Continue to step 4.

Are there any [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table) options?

**Yes.** Does the table match any of them?

**Yes.** Ignore the update and exit.

**No.** Continue to step 5.

**No.** Continue to step 5.

Are there any [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options?

**Yes.** Does the table match any of them?

**Yes.** Execute the update and exit.

**No.** Continue to step 6.

**No.** Continue to step 6.

Are there any [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) options?

**Yes.** Does the table match any of them?

**Yes.** Ignore the update and exit.

**No.** Continue to step 7.

**No.** Continue to step 7.

Is there another table to be tested?

**Yes.** Go back to step 3.

**No.** Continue to step 8.

Are there any [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) or [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options?

**Yes.** Ignore the update and exit.

**No.** Execute the update and exit.

**Note**

Statement-based replication stops if a single SQL statement operates on both a table that is included by a [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) or [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) option, and another table that is ignored by a [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table) or [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) option. The replica must either execute or ignore the complete statement (which forms a replication event), and it cannot logically do this. This also applies to row-based replication for DDL statements, because DDL statements are always logged as statements, without regard to the logging format in effect. The only type of statement that can update both an included and an ignored table and still be replicated successfully is a DML statement that has been logged with [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format).

**17.2.5.3 Interactions Between Replication Filtering Options**

If you use a combination of database-level and table-level replication filtering options, the replica first accepts or ignores events using the database options, then it evaluates all events permitted by those options according to the table options. This can sometimes lead to results that seem counterintuitive. It is also important to note that the results vary depending on whether the operation is logged using statement-based or row-based binary logging format. If you want to be sure that your replication filters always operate in the same way independently of the binary logging format, which is particularly important if you are using mixed binary logging format, follow the guidance in this topic.

The effect of the replication filtering options differs between binary logging formats because of the way the database name is identified. With statement-based format, DML statements are handled based on the current database, as specified by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement. With row-based format, DML statements are handled based on the database where the modified table exists. DDL statements are always filtered based on the current database, as specified by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement, regardless of the binary logging format.

An operation that involves multiple tables can also be affected differently by replication filtering options depending on the binary logging format. Operations to watch out for include transactions involving multi-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, triggers, cascading foreign keys, stored functions that update multiple tables, and DML statements that invoke stored functions that update one or more tables. If these operations update both filtered-in and filtered-out tables, the results can vary with the binary logging format.

If you need to guarantee that your replication filters operate consistently regardless of the binary logging format, particularly if you are using mixed binary logging format ([**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), use only table-level replication filtering options, and do not use database-level replication filtering options. Also, do not use multi-table DML statements that update both filtered-in and filtered-out tables.

If you need to use a combination of database-level and table-level replication filters, and want these to operate as consistently as possible, choose one of the following strategies:

If you use row-based binary logging format ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), for DDL statements, rely on the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement to set the database and do not specify the database name. You can consider changing to row-based binary logging format for improved consistency with replication filtering. See [Section 5.4.4.2, “Setting The Binary Log Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-setting) for the conditions that apply to changing the binary logging format.

If you use statement-based or mixed binary logging format ([**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) or **MIXED**), for both DML and DDL statements, rely on the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement and do not use the database name. Also, do not use multi-table DML statements that update both filtered-in and filtered-out tables.

**Example 17.7 A**[--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db)**option and a**[--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table)**option**

On the replication source server, the following statements are issued:

USE db1;

CREATE TABLE t2 LIKE t1;

INSERT INTO db2.t3 VALUES (1);

The replica has the following replication filtering options set:

replicate-ignore-db = db1

replicate-do-table = db2.t3

The DDL statement [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) creates the table in **db1**, as specified by the preceding [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement. The replica filters out this statement according to its [--replicate-ignore-db = db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) option, because **db1** is the current database. This result is the same whatever the binary logging format is on the replication source server. However, the result of the DML [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement is different depending on the binary logging format:

If row-based binary logging format is in use on the source ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), the replica evaluates the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operation using the database where the table exists, which is named as **db2**. The database-level option [--replicate-ignore-db = db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db), which is evaluated first, therefore does not apply. The table-level option [--replicate-do-table = db2.t3](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) does apply, so the replica applies the change to table **t3**.

If statement-based binary logging format is in use on the source ([**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), the replica evaluates the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operation using the default database, which was set by the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement to **db1** and has not been changed. According to its database-level [--replicate-ignore-db = db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) option, it therefore ignores the operation and does not apply the change to table **t3**. The table-level option [--replicate-do-table = db2.t3](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table) is not checked, because the statement already matched a database-level option and was ignored.

If the [--replicate-ignore-db = db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) option on the replica is necessary, and the use of statement-based (or mixed) binary logging format on the source is also necessary, the results can be made consistent by omitting the database name from the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement and relying on a [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement instead, as follows:

USE db1;

CREATE TABLE t2 LIKE t1;

USE db2;

INSERT INTO t3 VALUES (1);

In this case, the replica always evaluates the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement based on the database **db2**. Whether the operation is logged in statement-based or row-based binary format, the results remain the same.

**17.2.5.4 Replication Channel Based Filters**

This section explains how to work with replication filters when multiple replication channels exist, for example in a multi-source replication topology. Before MySQL 8.0, replication filters were global, so filters were applied to all replication channels. From MySQL 8.0, replication filters can be global or channel specific, enabling you to configure multi-source replicas with replication filters on specific replication channels. Channel specific replication filters are particularly useful in a multi-source replication topology when the same database or table is present on multiple sources, and the replica is only required to replicate it from one source.

For instructions to set up replication channels, see [Section 17.1.5, “MySQL Multi-Source Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-multi-source), and for more information on how they work, see [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels).

**Important**

Each channel on a multi-source replica must replicate from a different source. You cannot set up multiple replication channels from a single replica to a single source, even if you use replication filters to select different data to replicate on each channel. This is because the server IDs of replicas must be unique in a replication topology. The source distinguishes replicas only by their server IDs, not by the names of the replication channels, so it cannot recognize different replication channels from the same replica.

**Important**

On a MySQL server instance that is configured for Group Replication, channel specific replication filters can be used on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be used on the **group\_replication\_applier** or **group\_replication\_recovery** channels. Filtering on these channels would make the group unable to reach agreement on a consistent state.

**Important**

For a multi-source replica in a diamond topology (where the replica replicates from two or more sources, which in turn replicate from a common source), when GTID-based replication is in use, ensure that any replication filters or other channel configuration are identical on all channels on the multi-source replica. With GTID-based replication, filters are applied only to the transaction data, and GTIDs are not filtered out. This happens so that a replica’s GTID set stays consistent with the source’s, meaning GTID auto-positioning can be used without re-acquiring filtered out transactions each time. In the case where the downstream replica is multi-source and receives the same transaction from multiple sources in a diamond topology, the downstream replica now has multiple versions of the transaction, and the result depends on which channel applies the transaction first. The second channel to attempt it skips the transaction using GTID auto-skip, because the transaction’s GTID was added to the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) set by the first channel. With identical filtering on the channels, there is no problem because all versions of the transaction contain the same data, so the results are the same. However, with different filtering on the channels, the database can become inconsistent and replication can hang.

**Overview of Replication Filters and Channels**

When multiple replication channels exist, for example in a multi-source replication topology, replication filters are applied as follows:

Any global replication filter specified is added to the global replication filters of the filter type (**do\_db**, **do\_ignore\_table**, and so on).

Any channel specific replication filter adds the filter to the specified channel’s replication filters for the specified filter type.

Each replication channel copies global replication filters to its channel specific replication filters if no channel specific replication filter of this type is configured.

Each channel uses its channel specific replication filters to filter the replication stream.

The syntax to create channel specific replication filters extends the existing SQL statements and command options. When a replication channel is not specified the global replication filter is configured to ensure backwards compatibility. The [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement supports the **FOR CHANNEL** clause to configure channel specific filters online. The **--replicate-\*** command options to configure filters can specify a replication channel using the form **--replicate-*filter\_type*=*channel\_name*:*filter\_details***. For example, suppose channels **channel\_1** and **channel\_2** exist before the server starts, starting the replica with the command line options [--replicate-do-db=db1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) [--replicate-do-db=channel\_1:db2](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) [--replicate-do-db=db3](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) [--replicate-ignore-db=db4](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) [--replicate-ignore-db=channel\_2:db5](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) would result in:

*Global replication filters*: do\_db=db1,db3, ignore\_db=db4

*Channel specific filters on channel\_1*: do\_db=db2 ignore\_db=db4

*Channel specific filters on channel\_2*: do\_db=db1,db3 ignore\_db=db5

To monitor the replication filters in such a setup use the [**replication\_applier\_global\_filters**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-global-filters-table) and [**replication\_applier\_filters**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-filters-table) tables.

**Configuring Channel Specific Replication Filters at Startup**

The replication filter related command options can take an optional ***channel*** followed by a colon, followed by the filter specification. The first colon is interpreted as a separator, subsequent colons are interpreted as literal colons. The following command options support channel specific replication filters using this format:

**--replicate-do-db=*channel*:*database\_id***

**--replicate-ignore-db=*channel*:*database\_id***

**--replicate-do-table=*channel*:*table\_id***

**--replicate-ignore-table=*channel*:*table\_id***

**--replicate-rewrite-db=*channel*:*db1-db2***

**--replicate-wild-do-table=*channel*:*table regexid***

**--replicate-wild-ignore-table=*channel*:*table regexid***

If you use a colon but do not specify a ***channel*** for the filter option, for example --replicate-do-db=:***database\_id***, the option configures the replication filter for the default replication channel. The default replication channel is the replication channel which always exists once replication has been started, and differs from multi-source replication channels which you create manually. When neither the colon nor a ***channel*** is specified the option configures the global replication filters, for example --replicate-do-db=***database\_id*** configures the global [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) filter.

If you configure multiple **rewrite-db=*from\_name*->*to\_name*** options with the same ***from\_name*** database, all filters are added together (put into the **rewrite\_do** list) and the first one takes effect.

**Changing Channel Specific Replication Filters Online**

In addition to the --replicate-\* options, replication filters can be configured using the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement. This removes the need to restart the server, but the replication SQL thread must be stopped while making the change. To make this statement apply the filter to a specific channel, use the **FOR CHANNEL *channel*** clause. For example:

CHANGE REPLICATION FILTER REPLICATE\_DO\_DB=(db1) FOR CHANNEL channel\_1;

When a **FOR CHANNEL** clause is provided, the statement acts on the specified channel's replication filters. If multiple types of filters (**do\_db**, **do\_ignore\_table**, **wild\_do\_table**, and so on) are specified, only the specified filter types are replaced by the statement. In a replication topology with multiple channels, for example on a multi-source replica, when no **FOR CHANNEL** clause is provided, the statement acts on the global replication filters and all channels’ replication filters, using a similar logic as the **FOR CHANNEL** case. For more information see [Section 13.4.2.2, “CHANGE REPLICATION FILTER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

**Removing Channel Specific Replication Filters**

When channel specific replication filters have been configured, you can remove the filter by issuing an empty filter type statement. For example to remove all **REPLICATE\_REWRITE\_DB** filters from a replication channel named **channel\_1** issue:

CHANGE REPLICATION FILTER REPLICATE\_REWRITE\_DB=() FOR CHANNEL channel\_1;

Any **REPLICATE\_REWRITE\_DB** filters previously configured, using either command options or [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter), are removed.

The [**RESET REPLICA | SLAVE ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement removes channel specific replication filters that were set on channels deleted by the statement. When the deleted channel or channels are recreated, any global replication filters specified for the replica are copied to them, and no channel specific replication filters are applied.

**17.3 Replication Security**

[17.3.1 Setting Up Replication to Use Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections)

[17.3.2 Encrypting Binary Log Files and Relay Log Files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption)

[17.3.3 Replication Privilege Checks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)

To protect against unauthorized access to data that is stored on and transferred between replication source servers and replicas, set up all the servers involved using the security measures that you would choose for any MySQL instance in your installation, as described in [Chapter 6, *Security*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html). In addition, for servers in a replication topology, consider implementing the following security measures:

Set up sources and replicas to use encrypted connections to transfer the binary log, which protects this data in motion. Encryption for these connections must be activated using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, in addition to setting up the servers to support encrypted network connections. See [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections).

Encrypt the binary log files and relay log files on sources and replicas, which protects this data at rest, and also any data in use in the binary log cache. Binary log encryption is activated using the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable. See [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption).

Apply privilege checks to replication appliers, which help to secure replication channels against the unauthorized or accidental use of privileged or unwanted operations. Privilege checks are implemented by setting up a **PRIVILEGE\_CHECKS\_USER** account, which MySQL uses to verify that you have authorized each specific transaction for that channel. See [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks).

For Group Replication, binary log encryption and privilege checks can be used as a security measure on replication group members. You should also consider encrypting the connections between group members, comprising group communication connections and distributed recovery connections, and applying IP address allowlisting to exclude untrusted hosts. For information on these security measures specific to Group Replication, see [Section 18.6, “Group Replication Security”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-security).

**17.3.1 Setting Up Replication to Use Encrypted Connections**

To use an encrypted connection for the transfer of the binary log required during replication, both the source and the replica servers must support encrypted network connections. If either server does not support encrypted connections (because it has not been compiled or configured for them), replication through an encrypted connection is not possible.

Setting up encrypted connections for replication is similar to doing so for client/server connections. You must obtain (or create) a suitable security certificate that you can use on the source, and a similar certificate (from the same certificate authority) on each replica. You must also obtain suitable key files.

For more information on setting up a server and client for encrypted connections, see [Section 6.3.1, “Configuring MySQL to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#using-encrypted-connections).

To enable encrypted connections on the source, you must create or obtain suitable certificate and key files, and then add the following configuration parameters to the source's configuration within the **[mysqld]** section of the source's my.cnf file, changing the file names as necessary:

[mysqld]

ssl\_ca=cacert.pem

ssl\_cert=server-cert.pem

ssl\_key=server-key.pem

The paths to the files may be relative or absolute; we recommend that you always use complete paths for this purpose.

The configuration parameters are as follows:

[**ssl\_ca**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_ca): The path name of the Certificate Authority (CA) certificate file. ([**ssl\_capath**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_capath) is similar but specifies the path name of a directory of CA certificate files.)

[**ssl\_cert**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_cert): The path name of the server public key certificate file. This certificate can be sent to the client and authenticated against the CA certificate that it has.

[**ssl\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_ssl_key): The path name of the server private key file.

To enable encrypted connections on the replica, use the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). You can either name the replica's certificate and SSL private key files required for the encrypted connection in the **[client]** section of the replica's my.cnf file, or you can explicitly specify that information using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement.

To name the replica's certificate and key files using an option file, add the following lines to the **[client]** section of the replica's my.cnf file, changing the file names as necessary:

[client]

ssl-ca=cacert.pem

ssl-cert=client-cert.pem

ssl-key=client-key.pem

Restart the replica server, using the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable, to prevent the replica from connecting to the source. Use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) to specify the source configuration, and add the **MASTER\_SSL** option to connect using encryption:

mysql> **CHANGE MASTER TO**

-> **MASTER\_HOST='source\_hostname',**

-> **MASTER\_USER='repl',**

-> **MASTER\_PASSWORD='*password*',**

-> **MASTER\_SSL=1;**

Or from MySQL 8.0.23:

mysql> **CHANGE REPLICATION SOURCE TO**

-> **SOURCE\_HOST='source\_hostname',**

-> **SOURCE\_USER='repl',**

-> **SOURCE\_PASSWORD='*password*',**

-> **SOURCE\_SSL=1;**

Setting **SOURCE\_SSL=1** | **MASTER\_SSL=1** for a replication connection and then setting no further **SOURCE\_SSL\_*xxx*** | **MASTER\_SSL\_*xxx*** options corresponds to setting **--ssl-mode=REQUIRED** for the client, as described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). With **SOURCE\_SSL=1** | **MASTER\_SSL=1**, the connection attempt only succeeds if an encrypted connection can be established. A replication connection does not fall back to an unencrypted connection, so there is no setting corresponding to the **--ssl-mode=PREFERRED** setting for replication. If **SOURCE\_SSL=0** | **MASTER\_SSL=0** is set, this corresponds to **--ssl-mode=DISABLED**.

To name the replica's certificate and SSL private key files using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, if you did not do this in the replica's my.cnf file, add the appropriate **SOURCE\_SSL\_*xxx*** | **MASTER\_SSL\_*xxx*** options:

-> **MASTER\_SSL\_CA = 'ca\_file\_name',**

-> **MASTER\_SSL\_CAPATH = 'ca\_directory\_name',**

-> **MASTER\_SSL\_CERT = 'cert\_file\_name',**

-> **MASTER\_SSL\_KEY = 'key\_file\_name',**

These options correspond to the **--ssl-*xxx*** options with the same names, as described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). For these options to take effect, **MASTER\_SSL=1** must also be set. For a replication connection, specifying a value for either of **MASTER\_SSL\_CA** or **MASTER\_SSL\_CAPATH**, or specifying these options in the replica's my.cnf file, corresponds to setting **--ssl-mode=VERIFY\_CA**. The connection attempt only succeeds if a valid matching Certificate Authority (CA) certificate is found using the specified information.

To activate host name identity verification, add the **MASTER\_SSL\_VERIFY\_SERVER\_CERT** option:

-> **MASTER\_SSL\_VERIFY\_SERVER\_CERT=1,**

This option corresponds to the **--ssl-verify-server-cert** option, which was deprecated from MySQL 5.7 and removed in MySQL 8.0. For a replication connection, specifying **MASTER\_SSL\_VERIFY\_SERVER\_CERT=1** corresponds to setting **--ssl-mode=VERIFY\_IDENTITY**, as described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). For this option to take effect, **MASTER\_SSL=1** must also be set. Host name identity verification does not work with self-signed certificates.

To activate certificate revocation list (CRL) checks, add the **MASTER\_SSL\_CRL** or **MASTER\_SSL\_CRLPATH** option:

-> **MASTER\_SSL\_CRL = 'crl\_file\_name',**

-> **MASTER\_SSL\_CRLPATH = 'crl\_directory\_name',**

These options correspond to the **--ssl-*xxx*** options with the same names, as described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). If they are not specified, no CRL checking takes place.

To specify lists of ciphers, ciphersuites, and encryption protocols permitted by the replica for the replication connection, use the **MASTER\_SSL\_CIPHER**, **MASTER\_TLS\_VERSION**, and **MASTER\_TLS\_CIPHERSUITES** options:

-> **MASTER\_SSL\_CIPHER = 'cipher\_list',**

-> **MASTER\_TLS\_VERSION = 'protocol\_list',**

-> **MASTER\_TLS\_CIPHERSUITES = 'ciphersuite\_list',**

The **MASTER\_SSL\_CIPHER** option specifies a colon-separated list of one or more ciphers permitted by the replica for the replication connection.

The **MASTER\_TLS\_VERSION** option specifies a comma-separated list of the TLS encryption protocols permitted by the replica for the replication connection, in a format like that for the [**tls\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tls_version) server system variable. The connection procedure negotiates the use of the highest TLS version that both the source and the replica permit. To be able to connect, the replica must have at least one TLS version in common with the source.

The **MASTER\_TLS\_CIPHERSUITES** option (available from MySQL 8.0.19) specifies a colon-separated list of one or more ciphersuites that are permitted by the replica for the replication connection if TLSv1.3 is used for the connection. If this option is set to **NULL** when TLSv1.3 is used (which is the default if you do not set the option), the ciphersuites that are enabled by default are allowed. If you set the option to an empty string, no cipher suites are allowed, and TLSv1.3 is therefore not used.

The protocols, ciphers, and ciphersuites that you can specify in these lists depend on the SSL library used to compile MySQL. For information about the formats, the permitted values, and the defaults if you do not specify the options, see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers).

**Note**

In MySQL 8.0.16 through 8.0.18, MySQL supports TLSv1.3, but the **MASTER\_TLS\_CIPHERSUITES** option is not available. In these releases, if TLSv1.3 is used for connections between a source and replica, the source must permit the use of at least one TLSv1.3 ciphersuite that is enabled by default. From MySQL 8.0.19, you can use the option to specify any selection of ciphersuites, including only non-default ciphersuites if you want.

After the source information has been updated, start the replication process on the replica:

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **START REPLICA;**

You can use the [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement to confirm that an encrypted connection was established successfully.

Requiring encrypted connections on the replica does not ensure that the source requires encrypted connections from replicas. If you want to ensure that the source only accepts replicas that connect using encrypted connections, create a replication user account on the source using the **REQUIRE SSL** option, then grant that user the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege. For example:

mysql> **CREATE USER 'repl'@'%.example.com' IDENTIFIED BY '*password*'**

-> **REQUIRE SSL;**

mysql> **GRANT REPLICATION SLAVE ON \*.\***

-> **TO 'repl'@'%.example.com';**

If you have an existing replication user account on the source, you can add **REQUIRE SSL** to it with this statement:

mysql> **ALTER USER 'repl'@'%.example.com' REQUIRE SSL;**

**17.3.2 Encrypting Binary Log Files and Relay Log Files**

[17.3.2.1 Scope of Binary Log Encryption](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption-scope)

[17.3.2.2 Binary Log Encryption Keys](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption-encryption-keys)

[17.3.2.3 Binary Log Master Key Rotation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption-key-rotation)

From MySQL 8.0.14, binary log files and relay log files can be encrypted, helping to protect these files and the potentially sensitive data contained in them from being misused by outside attackers, and also from unauthorized viewing by users of the operating system where they are stored. The encryption algorithm used for the files, the AES (Advanced Encryption Standard) cipher algorithm, is built in to MySQL Server and cannot be configured.

You enable this encryption on a MySQL server by setting the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable to **ON**. **OFF** is the default. The system variable sets encryption on for binary log files and relay log files. Binary logging does not need to be enabled on the server to enable encryption, so you can encrypt the relay log files on a replica that has no binary log. To use encryption, a keyring component or plugin must be installed and configured to supply MySQL Server's keyring service. For instructions to do this, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring). Any supported keyring component or plugin can be used to store binary log encryption keys.

When you first start the server with encryption enabled, a new binary log encryption key is generated before the binary log and relay logs are initialized. This key is used to encrypt a file password for each binary log file (if the server has binary logging enabled) and relay log file (if the server has replication channels), and further keys generated from the file passwords are used to encrypt the data in the files. The binary log encryption key that is currently in use on the server is called the binary log master key. The two tier encryption key architecture means that the binary log master key can be rotated (replaced by a new master key) as required, and only the file password for each file needs to be re-encrypted with the new master key, not the whole file. Relay log files are encrypted for all channels, including new channels that are created after encryption is activated. The binary log index file and relay log index file are never encrypted.

If you activate encryption while the server is running, a new binary log encryption key is generated at that time. The exception is if encryption was active previously on the server and was then disabled, in which case the binary log encryption key that was in use before is used again. The binary log file and relay log files are rotated immediately, and file passwords for the new files and all subsequent binary log files and relay log files are encrypted using this binary log encryption key. Existing binary log files and relay log files still present on the server are not encrypted, but you can purge them if they are no longer needed.

If you deactivate encryption by changing the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable to **OFF**, the binary log file and relay log files are rotated immediately and all subsequent logging is unencrypted. Previously encrypted files are not automatically decrypted, but the server is still able to read them. The [**BINLOG\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-encryption-admin) privilege is required to activate or deactivate encryption while the server is running.

Encrypted and unencrypted binary log files can be distinguished using the magic number at the start of the file header for encrypted log files (**0xFD62696E**), which differs from that used for unencrypted log files (**0xFE62696E**). The [**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs) statement shows whether each binary log file is encrypted or unencrypted.

When binary log files have been encrypted, [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) cannot read them directly, but can read them from the server using the [--read-from-remote-server](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_read-from-remote-server) option. From MySQL 8.0.14, [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) returns a suitable error if you attempt to read an encrypted binary log file directly, but older versions of [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) do not recognise the file as a binary log file at all. If you back up encrypted binary log files using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), note that the copies of the files that are generated using [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) are stored in an unencrypted format.

Binary log encryption can be combined with binary log transaction compression (available as of MySQL 8.0.20). For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

**17.3.2.1 Scope of Binary Log Encryption**

When binary log encryption is active for a MySQL server instance, the encryption coverage is as follows:

Data at rest that is written to the binary log files and relay log files is encrypted from the point in time where encryption is started, using the two tier encryption architecture described above. Existing binary log files and relay log files that were present on the server when you started encryption are not encrypted. You can purge these files when they are no longer needed.

Data in motion in the replication event stream, which is sent to MySQL clients including [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), is decrypted for transmission, and should therefore be protected in transit by the use of connection encryption (see [Section 6.3, “Using Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connections) and [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections)).

Data in use that is held in the binary log transaction and statement caches during a transaction is in unencrypted format in the memory buffer that stores the cache. The data is written to a temporary file on disk if it exceeds the space available in the memory buffer. From MySQL 8.0.17, when binary log encryption is active on the server, temporary files used to hold the binary log cache are encrypted using AES-CTR (AES Counter mode) for stream encryption. Because the temporary files are volatile and tied to a single process, they are encrypted using single-tier encryption, using a randomly generated file password and initialization vector that exist only in memory and are never stored on disk or in the keyring. After each transaction is committed, the binary log cache is reset: the memory buffer is cleared, any temporary file used to hold the binary log cache is truncated, and a new file password and initialization vector are randomly generated for use with the next transaction. This reset also takes place when the server is restarted after a normal shutdown or an unexpected halt.

**Note**

If you use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) when [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, which is not recommended as the statement is considered unsafe for statement-based replication, a temporary file containing the data is created on the replica where the changes are applied. These temporary files are not encrypted when binary log encryption is active on the server. Use row-based or mixed binary logging format instead, which do not create the temporary files.

**17.3.2.2 Binary Log Encryption Keys**

The binary log encryption keys used to encrypt the file passwords for the log files are 256-bit keys that are generated specifically for each MySQL server instance using MySQL Server's keyring service (see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring)). The keyring service handles the creation, retrieval, and deletion of the binary log encryption keys. A server instance only creates and removes keys generated for itself, but it can read keys generated for other instances if they are stored in the keyring, as in the case of a server instance that has been cloned by file copying.

**Important**

The binary log encryption keys for a MySQL server instance must be included in your backup and recovery procedures, because if the keys required to decrypt the file passwords for current and retained binary log files or relay log files are lost, it might not be possible to start the server.

The format of binary log encryption keys in the keyring is as follows:

MySQLReplicationKey\_{UUID}\_{SEQ\_NO}

For example:

MySQLReplicationKey\_00508583-b5ce-11e8-a6a5-0010e0734796\_1

**{UUID}** is the true UUID generated by the MySQL server (the value of the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) system variable). **{SEQ\_NO}** is the sequence number for the binary log encryption key, which is incremented by 1 for each new key that is generated on the server.

The binary log encryption key that is currently in use on the server is called the binary log master key. The sequence number for the current binary log master key is stored in the keyring. The binary log master key is used to encrypt each new log file's file password, which is a randomly generated 32-byte file password specific to the log file that is used to encrypt the file data. The file password is encrypted using AES-CBC (AES Cipher Block Chaining mode) with the 256-bit binary log encryption key and a random initialization vector (IV), and is stored in the log file's file header. The file data is encrypted using AES-CTR (AES Counter mode) with a 256-bit key generated from the file password and a nonce also generated from the file password. It is technically possible to decrypt an encrypted file offline, if the binary log encryption key used to encrypt the file password is known, by using tools available in the OpenSSL cryptography toolkit.

If you use file copying to clone a MySQL server instance that has encryption active so its binary log files and relay log files are encrypted, ensure that the keyring is also copied, so that the clone server can read the binary log encryption keys from the source server. When encryption is activated on the clone server (either at startup or subsequently), the clone server recognizes that the binary log encryption keys used with the copied files include the generated UUID of the source server. It automatically generates a new binary log encryption key using its own generated UUID, and uses this to encrypt the file passwords for subsequent binary log files and relay log files. The copied files continue to be read using the source server's keys.

**17.3.2.3 Binary Log Master Key Rotation**

When binary log encryption is enabled, you can rotate the binary log master key at any time while the server is running by issuing [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key). When the binary log master key is rotated manually using this statement, the passwords for the new and subsequent files are encrypted using the new binary log master key, and also the file passwords for existing encrypted binary log files and relay log files are re-encrypted using the new binary log master key, so the encryption is renewed completely. You can rotate the binary log master key on a regular basis to comply with your organization's security policy, and also if you suspect that the current or any of the previous binary log master keys might have been compromised.

When you rotate the binary log master key manually, MySQL Server takes the following actions in sequence:

A new binary log encryption key is generated with the next available sequence number, stored on the keyring, and used as the new binary log master key.

The binary log and relay log files are rotated on all channels.

The new binary log master key is used to encrypt the file passwords for the new binary log and relay log files, and subsequent files until the key is changed again.

The file passwords for existing encrypted binary log files and relay log files on the server are re-encrypted in turn using the new binary log master key, starting with the most recent files. Any unencrypted files are skipped.

Binary log encryption keys that are no longer in use for any files after the re-encryption process are removed from the keyring.

The [**BINLOG\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-encryption-admin) privilege is required to issue [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key), and the statement cannot be used if the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable is set to **OFF**.

As the final step of the binary log master key rotation process, all binary log encryption keys that no longer apply to any retained binary log files or relay log files are cleaned up from the keyring. If a retained binary log file or relay log file cannot be initialized for re-encryption, the relevant binary log encryption keys are not deleted in case the files can be recovered in the future. For example, this might be the case if a file listed in a binary log index file is currently unreadable, or if a channel fails to initialize. If the server UUID changes, for example because a backup created using MySQL Enterprise Backup is used to set up a new replica, issuing [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) on the new server does not delete any earlier binary log encryption keys that include the original server UUID.

If any of the first four steps of the binary log master key rotation process cannot be completed correctly, an error message is issued explaining the situation and the consequences for the encryption status of the binary log files and relay log files. Files that were previously encrypted are always left in an encrypted state, but their file passwords might still be encrypted using an old binary log master key. If you see these errors, first retry the process by issuing [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) again. Then investigate the status of individual files to see what is blocking the process, especially if you suspect that the current or any of the previous binary log master keys might have been compromised.

If the final step of the binary log master key rotation process cannot be completed correctly, a warning message is issued explaining the situation. The warning message identifies whether the process could not clean up the auxiliary keys in the keyring for rotating the binary log master key, or could not clean up unused binary log encryption keys. You can choose to ignore the message as the keys are auxiliary keys or no longer in use, or you can issue [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) again to retry the process.

If the server stops and is restarted with binary log encryption still set to **ON** during the binary log master key rotation process, new binary log files and relay log files after the restart are encrypted using the new binary log master key. However, the re-encryption of existing files is not continued, so files that did not get re-encrypted before the server stopped are left encrypted using the previous binary log master key. To complete re-encryption and clean up unused binary log encryption keys, issue [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) again after the restart.

[**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) actions are not written to the binary log and are not executed on replicas. Binary log master key rotation can therefore be carried out in replication environments including a mix of MySQL versions. To schedule regular rotation of the binary log master key on all applicable source and replica servers, you can enable the MySQL Event Scheduler on each server and issue the [**ALTER INSTANCE ROTATE BINLOG MASTER KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-rotate-binlog-master-key) statement using a [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement. If you rotate the binary log master key because you suspect that the current or any of the previous binary log master keys might have been compromised, issue the statement on every applicable source and replica server. Issuing the statement on individual servers ensures that you can verify immediate compliance, even in the case of replicas that are lagging, belong to multiple replication topologies, or are not currently active in the replication topology but have binary log and relay log files.

The [**binlog\_rotate\_encryption\_master\_key\_at\_startup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rotate_encryption_master_key_at_startup) system variable controls whether the binary log master key is automatically rotated when the server is restarted. If this system variable is set to **ON**, a new binary log encryption key is generated and used as the new binary log master key whenever the server is restarted. If it is set to **OFF**, which is the default, the existing binary log master key is used again after the restart. When the binary log master key is rotated at startup, the file passwords for the new binary log and relay log files are encrypted using the new key. The file passwords for the existing encrypted binary log files and relay log files are not re-encrypted, so they remain encrypted using the old key, which remains available on the keyring.

**17.3.3 Replication Privilege Checks**

[17.3.3.1 Privileges For The Replication PRIVILEGE\_CHECKS\_USER Account](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks-account)

[17.3.3.2 Privilege Checks For Group Replication Channels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks-gr)

[17.3.3.3 Recovering From Failed Replication Privilege Checks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks-recover)

By default, MySQL replication (including Group Replication) does not carry out privilege checks when transactions that were already accepted by another server are applied on a replica or group member. From MySQL 8.0.18, you can create a user account with the appropriate privileges to apply the transactions that are normally replicated on a channel, and specify this as the **PRIVILEGE\_CHECKS\_USER** account for the replication applier, using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). MySQL then checks each transaction against the user account's privileges to verify that you have authorized the operation for that channel. The account can also be safely used by an administrator to apply or reapply transactions from [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output, for example to recover from a replication error on the channel.

The use of a **PRIVILEGE\_CHECKS\_USER** account helps secure a replication channel against the unauthorized or accidental use of privileged or unwanted operations. The **PRIVILEGE\_CHECKS\_USER** account provides an additional layer of security in situations such as these:

You are replicating between a server instance on your organization's network, and a server instance on another network, such as an instance supplied by a cloud service provider.

You want to have multiple on-premise or off-site deployments administered as separate units, without giving one administrator account privileges on all the deployments.

You want to have an administrator account that enables an administrator to perform only operations that are directly relevant to the replication channel and the databases it replicates, rather than having wide privileges on the server instance.

You can increase the security of a replication channel where privilege checks are applied by adding one or both of these options to the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement when you specify the **PRIVILEGE\_CHECKS\_USER** account for the channel:

The **REQUIRE\_ROW\_FORMAT** option (available from MySQL 8.0.19) makes the replication channel accept only row-based replication events. When **REQUIRE\_ROW\_FORMAT** is set, you must use row-based binary logging ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)) on the source server. In MySQL 8.0.18, **REQUIRE\_ROW\_FORMAT** is not available, but the use of row-based binary logging for secured replication channels is still strongly recommended. With statement-based binary logging, some administrator-level privileges might be required for the **PRIVILEGE\_CHECKS\_USER** account to execute transactions successfully.

The **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option (available from MySQL 8.0.20) makes the replication channel use its own policy for primary key checks. Setting **ON** means that primary keys are always required, and setting **OFF** means that primary keys are never required. The default setting, **STREAM**, sets the session value of the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable using the value that is replicated from the source for each transaction. When **PRIVILEGE\_CHECKS\_USER** is set, setting **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to either **ON** or **OFF** means that the user account does not need session administration level privileges to set restricted session variables, which are required to change the value of [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key). It also normalizes the behavior across replication channels for different sources.

You grant the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege to enable a user account to appear as the **PRIVILEGE\_CHECKS\_USER** for a replication applier thread, and to execute the internal-use [**BINLOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog) statements used by mysqlbinlog. The user name and host name for the **PRIVILEGE\_CHECKS\_USER** account must follow the syntax described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names), and the user must not be an anonymous user (with a blank user name) or the **CURRENT\_USER**. To create a new account, use [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user). To grant this account the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege, use the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement. For example, to create a user account **priv\_repl**, which can be used manually by an administrator from any host in the **example.com** domain, and requires an encrypted connection, issue the following statements:

mysql> SET sql\_log\_bin = 0;

mysql> CREATE USER 'priv\_repl'@'%.example.com' IDENTIFIED BY '***password***' REQUIRE SSL;

mysql> GRANT REPLICATION\_APPLIER ON \*.\* TO 'priv\_repl'@'%.example.com';

mysql> SET sql\_log\_bin = 1;

The **SET sql\_log\_bin** statements are used so that the account management statements are not added to the binary log and sent to the replication channels (see [Section 13.4.1.3, “SET sql\_log\_bin Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-sql-log-bin)).

**Important**

The **caching\_sha2\_password** authentication plugin is the default for new users created from MySQL 8.0 (for details, see [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication)). To connect to a server using a user account that authenticates with this plugin, you must either set up an encrypted connection as described in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections), or enable the unencrypted connection to support password exchange using an RSA key pair.

After setting up the user account, use the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement to grant additional privileges to enable the user account to make the database changes that you expect the applier thread to carry out, such as updating specific tables held on the server. These same privileges enable an administrator to use the account if they need to execute any of those transactions manually on the replication channel. If an unexpected operation is attempted for which you did not grant the appropriate privileges, the operation is disallowed and the replication applier thread stops with an error. [Section 17.3.3.1, “Privileges For The Replication PRIVILEGE\_CHECKS\_USER Account”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks-account) explains what additional privileges the account needs. For example, to grant the **priv\_repl** user account the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege to add rows to the **cust** table in **db1**, issue the following statement:

mysql> GRANT INSERT ON db1.cust TO 'priv\_repl'@'%.example.com';

You assign the **PRIVILEGE\_CHECKS\_USER** account for a replication channel using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). The use of row-based binary logging is strongly recommended when **PRIVILEGE\_CHECKS\_USER** is set, and from MySQL 8.0.19 you can use the statement to set **REQUIRE\_ROW\_FORMAT** to enforce this. If replication is running, issue [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) before the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, and [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) after it. For example, to start privilege checks on the channel **channel\_1** on a running replica, issue the following statements:

mysql> **STOP SLAVE FOR CHANNEL 'channel\_1';**

mysql> **CHANGE MASTER TO**

**PRIVILEGE\_CHECKS\_USER = 'priv\_repl'@'%.example.com',**

**REQUIRE\_ROW\_FORMAT = 1 FOR CHANNEL 'channel\_1';**

mysql> **START SLAVE FOR CHANNEL 'channel\_1';**

Or from MySQL 8.0.22 / 8.0.23:

mysql> **STOP REPLICA FOR CHANNEL 'channel\_1';**

mysql> **CHANGE REPLICATION SOURCE TO**

**PRIVILEGE\_CHECKS\_USER = 'priv\_repl'@'%.example.com',**

**REQUIRE\_ROW\_FORMAT = 1 FOR CHANNEL 'channel\_1';**

mysql> **START REPLICA FOR CHANNEL 'channel\_1';**

When you restart the replication channel, the privilege checks are applied from that point on. If you do not specify a channel and no other channels exist, the statement is applied to the default channel. The user name and host name for the **PRIVILEGE\_CHECKS\_USER** account for a channel are shown in the Performance Schema [**replication\_applier\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-configuration-table) table, where they are properly escaped so they can be copied directly into SQL statements to execute individual transactions.

When **REQUIRE\_ROW\_FORMAT** is set for a replication channel, the replication applier does not create or drop temporary tables, and so does not set the [**pseudo\_thread\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_pseudo_thread_id) session system variable. It does not execute **LOAD DATA INFILE** instructions, and so does not attempt file operations to access or delete the temporary files associated with data loads (logged as a **Format\_description\_log\_event**). It does not execute **INTVAR**, **RAND**, and **USER\_VAR** events, which are used to reproduce the client's connection state for statement-based replication. (An exception is **USER\_VAR** events that are associated with DDL queries, which are executed.) It does not execute any statements that are logged within DML transactions. If the replication applier detects any of these types of event while attempting to queue or apply a transaction, the event is not applied, and replication stops with an error.

You can set **REQUIRE\_ROW\_FORMAT** for a replication channel whether or not you set a **PRIVILEGE\_CHECKS\_USER** account. The restrictions implemented when you set this option increase the security of the replication channel even without privilege checks. You can also specify the --require-row-format option when you use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog), to enforce row-based replication events in [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output.

**Security Context.** By default, when a replication applier thread is started with a user account specified as the **PRIVILEGE\_CHECKS\_USER**, the security context is created using default roles, or with all roles if [**activate\_all\_roles\_on\_login**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_activate_all_roles_on_login) is set to **ON**.

You can use roles to supply a general privilege set to accounts that are used as **PRIVILEGE\_CHECKS\_USER** accounts, as in the following example. Here, instead of granting the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **db1.cust** table directly to a user account as in the earlier example, this privilege is granted to the role **priv\_repl\_role** along with the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege. The role is then used to grant the privilege set to two user accounts, both of which can now be used as **PRIVILEGE\_CHECKS\_USER** accounts:

mysql> **SET sql\_log\_bin = 0;**

mysql> **CREATE USER 'priv\_repa'@'%.example.com'**

**IDENTIFIED BY '*password*'**

**REQUIRE SSL;**

mysql> **CREATE USER 'priv\_repb'@'%.example.com'**

**IDENTIFIED BY '*password*'**

**REQUIRE SSL;**

mysql> **CREATE ROLE 'priv\_repl\_role';**

mysql> **GRANT REPLICATION\_APPLIER TO 'priv\_repl\_role';**

mysql> **GRANT INSERT ON db1.cust TO 'priv\_repl\_role';**

mysql> **GRANT 'priv\_repl\_role' TO**

**'priv\_repa'@'%.example.com',**

**'priv\_repb'@'%.example.com';**

mysql> **SET DEFAULT ROLE 'priv\_repl\_role' TO**

**'priv\_repa'@'%.example.com',**

**'priv\_repb'@'%.example.com';**

mysql> **SET sql\_log\_bin = 1;**

Be aware that when the replication applier thread creates the security context, it checks the privileges for the **PRIVILEGE\_CHECKS\_USER** account, but does not carry out password validation, and does not carry out checks relating to account management, such as checking whether the account is locked. The security context that is created remains unchanged for the lifetime of the replication applier thread.

**Limitation.** In MySQL 8.0.18 only, if the replica [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is restarted immediately after issuing a [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement (due to an unexpected server exit or deliberate restart), the **PRIVILEGE\_CHECKS\_USER** account setting, which is held in the **mysql.slave\_relay\_log\_info** table, is lost and must be respecified. When you use privilege checks in that release, always verify that they are in place after a restart, and respecify them if required. From MySQL 8.0.19, the **PRIVILEGE\_CHECKS\_USER** account setting is preserved in this situation, so it is retrieved from the table and reapplied to the channel.

**17.3.3.1 Privileges For The Replication PRIVILEGE\_CHECKS\_USER Account**

The user account that is specified using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement as the **PRIVILEGE\_CHECKS\_USER** account for a replication channel must have the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege, otherwise the replication applier thread does not start. As explained in [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks), the account requires further privileges that are sufficient to apply all the expected transactions expected on the replication channel. These privileges are checked only when relevant transactions are executed.

The use of row-based binary logging ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)) is strongly recommended for replication channels that are secured using a **PRIVILEGE\_CHECKS\_USER** account. With statement-based binary logging, some administrator-level privileges might be required for the **PRIVILEGE\_CHECKS\_USER** account to execute transactions successfully. From MySQL 8.0.19, the **REQUIRE\_ROW\_FORMAT** setting can be applied to secured channels, which restricts the channel from executing events that would require these privileges.

The [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege explicitly or implicitly allows the **PRIVILEGE\_CHECKS\_USER** account to carry out the following operations that a replication thread needs to perform:

Setting the value of the system variables [**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next), [**original\_commit\_timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_commit_timestamp), [**original\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_server_version), [**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version), and [**pseudo\_slave\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_pseudo_slave_mode), to apply appropriate metadata and behaviors when executing transactions.

Executing internal-use [**BINLOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog) statements to apply [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output, provided that the account also has permission for the tables and operations in those statements.

Updating the system tables **mysql.gtid\_executed**, **mysql.slave\_relay\_log\_info**, **mysql.slave\_worker\_info**, and **mysql.slave\_master\_info**, to update replication metadata. (If events access these tables explicitly for other purposes, you must grant the appropriate privileges on the tables.)

Applying a binary log **Table\_map\_log\_event**, which provides table metadata but does not make any database changes.

If the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement is set to the default of **STREAM**, the **PRIVILEGE\_CHECKS\_USER** account needs privileges sufficient to set restricted session variables, so that it can change the value of the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable for the duration of a session to match the setting replicated from the source. The [**SESSION\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_session-variables-admin) privilege gives the account this capability. This privilege also allows the account to apply [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output that was created using the [--disable-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqlbinlog_disable-log-bin) option. If you set **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to either **ON** or **OFF**, the replica always uses that value for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable in replication operations, and so does not need these session administration level privileges.

If table encryption is in use, the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) system variable is set to **ON**, and the encryption setting for the tablespace involved in any event differs from the applying server's default encryption setting (specified by the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) system variable), the **PRIVILEGE\_CHECKS\_USER** account needs the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege in order to override the default encryption setting. It is strongly recommended that you do not grant this privilege. Instead, ensure that the default encryption setting on a replica matches the encryption status of the tablespaces that it replicates, and that replication group members have the same default encryption setting, so that the privilege is not needed.

In order to execute specific replicated transactions from the relay log, or transactions from [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output as required, the **PRIVILEGE\_CHECKS\_USER** account must have the following privileges:

For a row insertion logged in row format (which are logged as a **Write\_rows\_log\_event**), the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege on the relevant table.

For a row update logged in row format (which are logged as an **Update\_rows\_log\_event**), the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege on the relevant table.

For a row deletion logged in row format (which are logged as a **Delete\_rows\_log\_event**), the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege on the relevant table.

If statement-based binary logging is in use (which is not recommended with a **PRIVILEGE\_CHECKS\_USER** account), for a transaction control statement such as **BEGIN** or **COMMIT** or DML logged in statement format (which are logged as a **Query\_log\_event**), the **PRIVILEGE\_CHECKS\_USER** account needs privileges to execute the statement contained in the event.

If [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) operations need to be carried out on the replication channel, use row-based binary logging ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)). With this logging format, the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege is not needed to execute the event, so do not give the **PRIVILEGE\_CHECKS\_USER** account this privilege. The use of row-based binary logging is strongly recommended with replication channels that are secured using a **PRIVILEGE\_CHECKS\_USER** account. If **REQUIRE\_ROW\_FORMAT** is set for the channel, row-based binary logging is required. The **Format\_description\_log\_event**, which deletes any temporary files created by [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) events, is processed without privilege checks. For more information, see [Section 17.5.1.19, “Replication and LOAD DATA”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-load-data).

If the [**init\_slave**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_init_slave) system variable is set to specify one or more SQL statements to be executed when the replication SQL thread starts, the **PRIVILEGE\_CHECKS\_USER** account must have the privileges needed to execute these statements.

It is recommended that you never give any ACL privileges to the **PRIVILEGE\_CHECKS\_USER** account, including [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user), [**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-role), [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop-role), and [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option), and do not permit the account to update the **mysql.user** table. With these privileges, the account could be used to create or modify user accounts on the server. To avoid ACL statements issued on the source server being replicated to the secured channel for execution (where they fail in the absence of these privileges), you can issue **SET sql\_log\_bin = 0** before all ACL statements and **SET sql\_log\_bin = 1** after them, to omit the statements from the source's binary log. Alternatively, you can set a dedicated current database before executing all ACL statements, and use a replication filter ([--binlog-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_binlog-ignore-db)) to filter out this database on the replica.

**17.3.3.2 Privilege Checks For Group Replication Channels**

From MySQL 8.0.19, as well as securing asynchronous and semi-synchronous replication, you may choose to use a **PRIVILEGE\_CHECKS\_USER** account to secure the two replication applier threads used by Group Replication. The **group\_replication\_applier** thread on each group member is used for applying the group's transactions, and the **group\_replication\_recovery** thread on each group member is used for state transfer from the binary log as part of distributed recovery when the member joins or rejoins the group.

To secure one of these threads, stop Group Replication, then issue the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) with the **PRIVILEGE\_CHECKS\_USER** option, specifying **group\_replication\_applier** or **group\_replication\_recovery** as the channel name. For example:

mysql> **STOP GROUP\_REPLICATION;**

mysql> **CHANGE MASTER TO PRIVILEGE\_CHECKS\_USER = 'gr\_repl'@'%.example.com'**

**FOR CHANNEL 'group\_replication\_recovery';**

mysql> **START GROUP\_REPLICATION;**

Or from MySQL 8.0.23:

mysql> **STOP GROUP\_REPLICATION;**

mysql> **CHANGE REPLICATION SOURCE TO PRIVILEGE\_CHECKS\_USER = 'gr\_repl'@'%.example.com'**

**FOR CHANNEL 'group\_replication\_recovery';**

mysql> **START GROUP\_REPLICATION;**

For Group Replication channels, the **REQUIRE\_ROW\_FORMAT** setting is automatically enabled when the channel is created, and cannot be disabled, so you do not need to specify this.

**Important**

In MySQL 8.0.19, ensure that you do not issue the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement with the **PRIVILEGE\_CHECKS\_USER** option while Group Replication is running. This action causes the relay log files for the channel to be purged, which might cause the loss of transactions that have been received and queued in the relay log, but not yet applied.

Group Replication requires every table that is to be replicated by the group to have a defined primary key, or primary key equivalent where the equivalent is a non-null unique key. Rather than using the checks carried out by the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable, Group Replication has its own built-in set of checks for primary keys or primary key equivalents. You may set the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement to **ON** for a Group Replication channel. However, be aware that you might find some transactions that are permitted under Group Replication's built-in checks are not permitted under the checks carried out when you set **sql\_require\_primary\_key = ON** or **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK = ON**. For this reason, new and upgraded Group Replication channels from MySQL 8.0.20 (when the option was introduced) have **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** set to the default of **STREAM**, rather than to **ON**.

If a remote cloning operation is used for distributed recovery in Group Replication (see [Section 18.5.3.2, “Cloning for Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-cloning)), from MySQL 8.0.19, the **PRIVILEGE\_CHECKS\_USER** account and related settings from the donor are cloned to the joining member. If the joining member is set to start Group Replication on boot, it automatically uses the account for privilege checks on the appropriate replication channels.

In MySQL 8.0.18, due to a number of limitations, it is recommended that you do not use a **PRIVILEGE\_CHECKS\_USER** account with Group Replication channels.

**17.3.3.3 Recovering From Failed Replication Privilege Checks**

If a privilege check against the **PRIVILEGE\_CHECKS\_USER** account fails, the transaction is not executed and replication stops for the channel. Details of the error and the last applied transaction are recorded in the Performance Schema [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) table. Follow this procedure to recover from the error:

Identify the replicated event that caused the error and verify whether or not the event is expected and from a trusted source. You can use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to retrieve and display the events that were logged around the time of the error. For instructions to do this, see [Section 7.5, “Point-in-Time (Incremental) Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html#point-in-time-recovery).

If the replicated event is not expected or is not from a known and trusted source, investigate the cause. If you can identify why the event took place and there are no security considerations, proceed to fix the error as described below.

If the **PRIVILEGE\_CHECKS\_USER** account should have been permitted to execute the transaction, but has been misconfigured, grant the missing privileges to the account and restart replication for the channel.

If the transaction needs to be executed and you have verified that it is trusted, but the **PRIVILEGE\_CHECKS\_USER** account should not have this privilege normally, you can grant the required privilege to the **PRIVILEGE\_CHECKS\_USER** account temporarily. After the replicated event has been applied, remove the privilege from the account, and take any necessary steps to ensure the event does not recur if it is avoidable.

If the transaction is an administrative action that should only have taken place on the source and not on the replica, or should only have taken place on a single replication group member, skip the transaction on the server or servers where it stopped replication, then issue [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) to restart replication on the channel. To avoid the situation in future, you could issue such administrative statements with **SET sql\_log\_bin = 0** before them and **SET sql\_log\_bin = 1** after them, so that they are not logged on the source.

If the transaction is a DDL or DML statement that should not have taken place on either the source or the replica, skip the transaction on the server or servers where it stopped replication, undo the transaction manually on the server where it originally took place, then issue [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) to restart replication.

To skip a transaction, if GTIDs are in use, commit an empty transaction that has the GTID of the failing transaction, for example:

SET GTID\_NEXT='aaa-bbb-ccc-ddd:N';

BEGIN;

COMMIT;

SET GTID\_NEXT='AUTOMATIC';

If GTIDs are not in use, issue a **SET GLOBAL sql\_slave\_skip\_counter** statement to skip the event. For instructions to use this alternative method and more details about skipping transactions, see [Section 17.1.7.3, “Skipping Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-skip).

**17.4 Replication Solutions**

[17.4.1 Using Replication for Backups](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups)

[17.4.2 Handling an Unexpected Halt of a Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt)

[17.4.3 Monitoring Row-based Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-rbr-monitoring)

[17.4.4 Using Replication with Different Source and Replica Storage Engines](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-diffengines)

[17.4.5 Using Replication for Scale-Out](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-scaleout)

[17.4.6 Replicating Different Databases to Different Replicas](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-partitioning)

[17.4.7 Improving Replication Performance](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-performance)

[17.4.8 Switching Sources During Failover](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-switch)

[17.4.9 Switching Sources with Asynchronous Connection Failover](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover)

[17.4.10 Semisynchronous Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-semisync)

[17.4.11 Delayed Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-delayed)

Replication can be used in many different environments for a range of purposes. This section provides general notes and advice on using replication for specific solution types.

For information on using replication in a backup environment, including notes on the setup, backup procedure, and files to back up, see [Section 17.4.1, “Using Replication for Backups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups).

For advice and tips on using different storage engines on the source and replica, see [Section 17.4.4, “Using Replication with Different Source and Replica Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-diffengines).

Using replication as a scale-out solution requires some changes in the logic and operation of applications that use the solution. See [Section 17.4.5, “Using Replication for Scale-Out”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-scaleout).

For performance or data distribution reasons, you may want to replicate different databases to different replicas. See [Section 17.4.6, “Replicating Different Databases to Different Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-partitioning)

As the number of replicas increases, the load on the source can increase and lead to reduced performance (because of the need to replicate the binary log to each replica). For tips on improving your replication performance, including using a single secondary server as the source, see [Section 17.4.7, “Improving Replication Performance”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-performance).

For guidance on switching sources, or converting replicas into sources as part of an emergency failover solution, see [Section 17.4.8, “Switching Sources During Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-switch).

For information on security measures specific to servers in a replication topology, see [Section 17.3, “Replication Security”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-security).

**17.4.1 Using Replication for Backups**

[17.4.1.1 Backing Up a Replica Using mysqldump](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-mysqldump)

[17.4.1.2 Backing Up Raw Data from a Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-rawdata)

[17.4.1.3 Backing Up a Source or Replica by Making It Read Only](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-read-only)

To use replication as a backup solution, replicate data from the source to a replica, and then back up the replica. The replica can be paused and shut down without affecting the running operation of the source, so you can produce an effective snapshot of “live” data that would otherwise require the source to be shut down.

How you back up a database depends on its size and whether you are backing up only the data, or the data and the replica state so that you can rebuild the replica in the event of failure. There are therefore two choices:

If you are using replication as a solution to enable you to back up the data on the source, and the size of your database is not too large, the [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) tool may be suitable. See [Section 17.4.1.1, “Backing Up a Replica Using mysqldump”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-mysqldump).

For larger databases, where [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) would be impractical or inefficient, you can back up the raw data files instead. Using the raw data files option also means that you can back up the binary and relay logs that make it possible to re-create the replica in the event of a replica failure. For more information, see [Section 17.4.1.2, “Backing Up Raw Data from a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-rawdata).

Another backup strategy, which can be used for either source or replica servers, is to put the server in a read-only state. The backup is performed against the read-only server, which then is changed back to its usual read/write operational status. See [Section 17.4.1.3, “Backing Up a Source or Replica by Making It Read Only”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-backups-read-only).

**17.4.1.1 Backing Up a Replica Using mysqldump**

Using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to create a copy of a database enables you to capture all of the data in the database in a format that enables the information to be imported into another instance of MySQL Server (see [Section 4.5.4, “mysqldump — A Database Backup Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump)). Because the format of the information is SQL statements, the file can easily be distributed and applied to running servers in the event that you need access to the data in an emergency. However, if the size of your data set is very large, [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) may be impractical.

Tip

Consider using the [MySQL Shell dump utilities](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-dump-instance-schema.html), which provide parallel dumping with multiple threads, file compression, and progress information display, as well as cloud features such as Oracle Cloud Infrastructure Object Storage streaming, and MySQL Database Service compatibility checks and modifications. Dumps can be easily imported into a MySQL Server instance or a MySQL Database Service DB System using the [MySQL Shell load dump utilities](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-utilities-load-dump.html). Installation instructions for MySQL Shell can be found [here](https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-install.html).

When using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump), you should stop replication on the replica before starting the dump process to ensure that the dump contains a consistent set of data:

Stop the replica from processing requests. You can stop replication completely on the replica using [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin):

shell> **mysqladmin stop-slave**

Alternatively, you can stop only the replication SQL thread to pause event execution:

shell> **mysql -e 'STOP SLAVE SQL\_THREAD;'**

Or from MySQL 8.0.22:

shell> **mysql -e 'STOP REPLICA SQL\_THREAD;'**

This enables the replica to continue to receive data change events from the source's binary log and store them in the relay logs using the replication I/O thread, but prevents the replica from executing these events and changing its data. Within busy replication environments, permitting the replication I/O thread to run during backup may speed up the catch-up process when you restart the replication SQL thread.

Run [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to dump your databases. You may either dump all databases or select databases to be dumped. For example, to dump all databases:

shell> **mysqldump --all-databases > fulldb.dump**

Once the dump has completed, start replication again:

shell> **mysqladmin start-slave**

In the preceding example, you may want to add login credentials (user name, password) to the commands, and bundle the process up into a script that you can run automatically each day.

If you use this approach, make sure you monitor the replication process to ensure that the time taken to run the backup does not affect the replica's ability to keep up with events from the source. See [Section 17.1.7.1, “Checking Replication Status”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-status). If the replica is unable to keep up, you may want to add another replica and distribute the backup process. For an example of how to configure this scenario, see [Section 17.4.6, “Replicating Different Databases to Different Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-partitioning).

**17.4.1.2 Backing Up Raw Data from a Replica**

To guarantee the integrity of the files that are copied, backing up the raw data files on your MySQL replica should take place while your replica server is shut down. If the MySQL server is still running, background tasks may still be updating the database files, particularly those involving storage engines with background processes such as **InnoDB**. With **InnoDB**, these problems should be resolved during crash recovery, but since the replica server can be shut down during the backup process without affecting the execution of the source it makes sense to take advantage of this capability.

To shut down the server and back up the files:

Shut down the replica MySQL server:

shell> **mysqladmin shutdown**

Copy the data files. You can use any suitable copying or archive utility, including **cp**, **tar** or **WinZip**. For example, assuming that the data directory is located under the current directory, you can archive the entire directory as follows:

shell> **tar cf /tmp/dbbackup.tar ./data**

Start the MySQL server again. Under Unix:

shell> **mysqld\_safe &**

Under Windows:

C:\> **"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"**

Normally you should back up the entire data directory for the replica MySQL server. If you want to be able to restore the data and operate as a replica (for example, in the event of failure of the replica), in addition to the data, you need to have the replica's connection metadata repository and applier metadata repository, and the relay log files. These items are needed to resume replication after you restore the replica's data. Assuming tables have been used for the replica's connection metadata repository and applier metadata repository (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)), which is the default in MySQL 8.0, these tables are backed up along with the data directory. If files have been used for the repositories, which is deprecated, you must back these up separately. The relay log files must be backed up separately if they have been placed in a different location to the data directory.

If you lose the relay logs but still have the relay-log.info file, you can check it to determine how far the replication SQL thread has executed in the source's binary logs. Then you can use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) with the **SOURCE\_LOG\_FILE** | **MASTER\_LOG\_FILE** and **SOURCE\_LOG\_POS** | **MASTER\_LOG\_POS** options to tell the replica to re-read the binary logs from that point. This requires that the binary logs still exist on the source server.

If your replica is replicating [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements, you should also back up any SQL\_LOAD-\* files that exist in the directory that the replica uses for this purpose. The replica needs these files to resume replication of any interrupted [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) operations. The location of this directory is the value of the [**slave\_load\_tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_load_tmpdir) system variable. If the server was not started with that variable set, the directory location is the value of the [**tmpdir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmpdir) system variable.

**17.4.1.3 Backing Up a Source or Replica by Making It Read Only**

It is possible to back up either source or replica servers in a replication setup by acquiring a global read lock and manipulating the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable to change the read-only state of the server to be backed up:

Make the server read-only, so that it processes only retrievals and blocks updates.

Perform the backup.

Change the server back to its normal read/write state.

**Note**

The instructions in this section place the server to be backed up in a state that is safe for backup methods that get the data from the server, such as [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) (see [Section 4.5.4, “mysqldump — A Database Backup Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump)). You should not attempt to use these instructions to make a binary backup by copying files directly because the server may still have modified data cached in memory and not flushed to disk.

The following instructions describe how to do this for a source and for a replica. For both scenarios discussed here, suppose that you have the following replication setup:

A source server S1

A replica server R1 that has S1 as its source

A client C1 connected to S1

A client C2 connected to R1

In either scenario, the statements to acquire the global read lock and manipulate the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) variable are performed on the server to be backed up and do not propagate to any replicas of that server.

***Scenario 1: Backup with a Read-Only Source***

Put the source S1 in a read-only state by executing these statements on it:

mysql> **FLUSH TABLES WITH READ LOCK;**

mysql> **SET GLOBAL read\_only = ON;**

While S1 is in a read-only state, the following properties are true:

Requests for updates sent by C1 to S1 block because the server is in read-only mode.

Requests for query results sent by C1 to S1 succeed.

Making a backup on S1 is safe.

Making a backup on R1 is not safe. This server is still running, and might be processing the binary log or update requests coming from client C2.

While S1 is read only, perform the backup. For example, you can use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

After the backup operation on S1 completes, restore S1 to its normal operational state by executing these statements:

mysql> **SET GLOBAL read\_only = OFF;**

mysql> **UNLOCK TABLES;**

Although performing the backup on S1 is safe (as far as the backup is concerned), it is not optimal for performance because clients of S1 are blocked from executing updates.

This strategy applies to backing up a source in a replication setup, but can also be used for a single server in a nonreplication setting.

***Scenario 2: Backup with a Read-Only Replica***

Put the replica R1 in a read-only state by executing these statements on it:

mysql> **FLUSH TABLES WITH READ LOCK;**

mysql> **SET GLOBAL read\_only = ON;**

While R1 is in a read-only state, the following properties are true:

The source S1 continues to operate, so making a backup on the source is not safe.

The replica R1 is stopped, so making a backup on the replica R1 is safe.

These properties provide the basis for a popular backup scenario: Having one replica busy performing a backup for a while is not a problem because it does not affect the entire network, and the system is still running during the backup. In particular, clients can still perform updates on the source server, which remains unaffected by backup activity on the replica.

While R1 is read only, perform the backup. For example, you can use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

After the backup operation on R1 completes, restore R1 to its normal operational state by executing these statements:

mysql> **SET GLOBAL read\_only = OFF;**

mysql> **UNLOCK TABLES;**

After the replca is restored to normal operation, it again synchronizes to the source by catching up with any outstanding updates from the source's binary log.

**17.4.2 Handling an Unexpected Halt of a Replica**

In order for replication to be resilient to unexpected halts of the server (sometimes described as crash-safe) it must be possible for the replica to recover its state before halting. This section describes the impact of an unexpected halt of a replica during replication, and how to configure a replica for the best chance of recovery to continue replication.

After an unexpected halt of a replica, upon restart the replication SQL thread must recover information about which transactions have been executed already. The information required for recovery is stored in the replica's applier metadata repository. From MySQL 8.0, this repository is created by default as an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table named **mysql.slave\_relay\_log\_info**. By using this transactional storage engine the information is always recoverable upon restart. Updates to the applier metadata repository are committed together with the transactions, meaning that the replica's progress information recorded in that repository is always consistent with what has been applied to the database, even in the event of an unexpected server halt. For more information on the applier metadata repository, see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs).

DML transactions and also atomic DDL update the replication positions in the replica's applier metadata repository in the **mysql.slave\_relay\_log\_info** table together with applying the changes to the database, as an atomic operation. In all other cases, including DDL statements that are not fully atomic, and exempted storage engines that do not support atomic DDL, the **mysql.slave\_relay\_log\_info** table might be missing updates associated with replicated data if the server halts unexpectedly. Restoring updates in this case is a manual process. For details on atomic DDL support in MySQL 8.0, and the resulting behavior for the replication of certain statements, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

The recovery process by which a replica recovers from an unexpected halt varies depending on the configuration of the replica. The details of the recovery process are influenced by the chosen method of replication, whether the replica is single-threaded or multithreaded, and the setting of relevant system variables. The overall aim of the recovery process is to identify what transactions had already been applied on the replica's database before the unexpected halt occurred, and retrieve and apply the transactions that the replica missed following the unexpected halt.

For GTID-based replication, the recovery process needs the GTIDs of the transactions that were already received or committed by the replica. The missing transactions can be retrieved from the source using GTID auto-positioning, which automatically compares the source's transactions to the replica's transactions and identifies the missing transactions.

For file position based replication, the recovery process needs an accurate replication SQL thread (applier) position showing the last transaction that was applied on the replica. Based on that position, the replication I/O thread (receiver) retrieves from the source's binary log all of the transactions that should be applied on the replica from that point on.

Using GTID-based replication makes it easiest to configure replication to be resilient to unexpected halts. GTID auto-positioning means the replica can reliably identify and retrieve missing transactions, even if there are gaps in the sequence of applied transactions.

The following information provides combinations of settings that are appropriate for different types of replica to guarantee recovery as far as this is under the control of replication.

**Important**

Some factors outside the control of replication can have an impact on the replication recovery process and the overall state of replication after the recovery process. In particular, the settings that influence the recovery process for individual storage engines might result in transactions being lost in the event of an unexpected halt of a replica, and therefore unavailable to the replication recovery process. The [**innodb\_flush\_log\_at\_trx\_commit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) setting mentioned in the list below is a key setting for a replication setup that uses [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) with transactions. However, other settings specific to [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) or to other storage engines, especially those relating to flushing or synchronization, can also have an impact. Always check for and apply recommendations made by your chosen storage engines about crash-safe settings.

The following combination of settings on a replica is the most resilient to unexpected halts:

When GTID-based replication is in use ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), set **SOURCE\_AUTO\_POSITION=1** | **MASTER\_AUTO\_POSITION=1**, which activates GTID auto-positioning for the connection to the source to automatically identify and retrieve missing transactions. This option is set using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). If the replica has multiple replication channels, you need to set this option for each channel individually. For details of how GTID auto-positioning works, see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning). When file position based replication is in use, **SOURCE\_AUTO\_POSITION=1** | **MASTER\_AUTO\_POSITION=1** is not used, and instead the binary log position or relay log position is used to control where replication starts.

Set [**sync\_relay\_log=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log), which instructs the replication I/O thread to synchronize the relay log to disk after each received transaction is written to it. This means the replica's record of the current position read from the source's binary log (in the applier metadata repository) is never ahead of the record of transactions saved in the relay log. Note that although this setting is the safest, it is also the slowest due to the number of disk writes involved. With **sync\_relay\_log > 1**, or **sync\_relay\_log=0** (where synchronization is handled by the operating system), in the event of an unexpected halt of a replica there might be committed transactions that have not been synchronized to disk. Such transactions can cause the recovery process to fail if the recovering replica, based on the information it has in the relay log as last synchronized to disk, tries to retrieve and apply the transactions again instead of skipping them. Setting **sync\_relay\_log=1** is particularly important for a multi-threaded replica, where the recovery process fails if gaps in the sequence of transactions cannot be filled using the information in the relay log. For a single-threaded replica, the recovery process only needs to use the relay log if the relevant information is not available in the applier metadata repository.

Set [**innodb\_flush\_log\_at\_trx\_commit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit), which synchronizes the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) logs to disk before each transaction is committed. This setting, which is the default, ensures that [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables and the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) logs are saved on disk so that there is no longer a requirement for the information in the relay log regarding the transaction. Combined with the setting [**sync\_relay\_log=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log), this setting further ensures that the content of the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables and the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) logs is consistent with the content of the relay log at all times, so that purging the relay log files cannot cause unfillable gaps in the replica's history of transactions in the event of an unexpected halt.

Set [**relay\_log\_info\_repository = TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository), which stores the replication SQL thread position in the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table **mysql.slave\_relay\_log\_info**, and updates it together with the transaction commit to ensure a record that is always accurate. This setting is the default from MySQL 8.0, and the **FILE** setting is deprecated. From MySQL 8.0.23, the use of the system variable itself is deprecated, so omit it and allow it to default. If the **FILE** setting is used, which was the default in earlier releases, the information is stored in a file in the data directory that is updated after the transaction has been applied. This creates a risk of losing synchrony with the source depending at which stage of processing a transaction the replica halts at, or even corruption of the file itself. With the setting [**relay\_log\_info\_repository = FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository), recovery is not guaranteed.

Set [**relay\_log\_recovery = ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery), which enables automatic relay log recovery immediately following server startup. This global variable defaults to **OFF** and is read-only at runtime, but you can set it to **ON** with the [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) option at replica startup following an unexpected halt of a replica. Note that this setting ignores the existing relay log files, in case they are corrupted or inconsistent. The relay log recovery process starts a new relay log file and fetches transactions from the source beginning at the replication SQL thread position recorded in the applier metadata repository. The previous relay log files are removed over time by the replica's normal purge mechanism.

For a multithreaded replica, setting [**relay\_log\_recovery = ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) automatically handles any inconsistencies and gaps in the sequence of transactions that have been executed from the relay log. These gaps can occur when file position based replication is in use. (For more details, see [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies).) The relay log recovery process deals with gaps using the same method as the [**START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement would. When the replica reaches a consistent gap-free state, the relay log recovery process goes on to fetch further transactions from the source beginning at the replication SQL thread position. When GTID-based replication is in use, from MySQL 8.0.18 a multithreaded replica checks first whether **MASTER\_AUTO\_POSITION** is set to **ON**, and if it is, omits the step of calculating the transactions that should be skipped or not skipped, so that the old relay logs are not required for the recovery process.

**17.4.3 Monitoring Row-based Replication**

The current progress of the replication applier (SQL) thread when using row-based replication is monitored through Performance Schema instrument stages, enabling you to track the processing of operations and check the amount of work completed and work estimated. When these Performance Schema instrument stages are enabled the [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table shows stages for applier threads and their progress. For background information, see [Section 27.12.5, “Performance Schema Stage Event Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-stage-tables).

To track progress of all three row-based replication event types (write, update, delete):

Enable the three Performance Schema stages by issuing:

mysql> **UPDATE performance\_schema.setup\_instruments SET ENABLED = 'YES'**

-> **WHERE NAME LIKE 'stage/sql/Applying batch of row changes%';**

Wait for some events to be processed by the replication applier thread and then check progress by looking into the [**events\_stages\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-stages-current-table) table. For example to get progress for **update** events issue:

mysql> **SELECT WORK\_COMPLETED, WORK\_ESTIMATED FROM performance\_schema.events\_stages\_current**

-> **WHERE EVENT\_NAME LIKE 'stage/sql/Applying batch of row changes (update)'**

If [**binlog\_rows\_query\_log\_events**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_rows_query_log_events) is enabled, information about queries is stored in the binary log and is exposed in the **processlist\_info** field. To see the original query that triggered this event:

mysql> **SELECT db, processlist\_state, processlist\_info FROM performance\_schema.threads**

-> **WHERE processlist\_state LIKE 'stage/sql/Applying batch of row changes%' AND thread\_id = N;**

**17.4.4 Using Replication with Different Source and Replica Storage Engines**

It does not matter for the replication process whether the original table on the source and the replicated table on the replica use different storage engine types. In fact, the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) system variable is not replicated.

This provides a number of benefits in the replication process in that you can take advantage of different engine types for different replication scenarios. For example, in a typical scale-out scenario (see [Section 17.4.5, “Using Replication for Scale-Out”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-scaleout)), you want to use **InnoDB** tables on the source to take advantage of the transactional functionality, but use **MyISAM** on the replicas where transaction support is not required because the data is only read. When using replication in a data-logging environment you may want to use the **Archive** storage engine on the replica.

Configuring different engines on the source and replica depends on how you set up the initial replication process:

If you used [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to create the database snapshot on your source, you could edit the dump file text to change the engine type used on each table.

Another alternative for [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) is to disable engine types that you do not want to use on the replica before using the dump to build the data on the replica. For example, you can add the [--skip-federated](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#option_mysqld_innodb) option on your replica to disable the **FEDERATED** engine. If a specific engine does not exist for a table to be created, MySQL uses the default engine type, usually **InnoDB**. (This requires that the [**NO\_ENGINE\_SUBSTITUTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_engine_substitution) SQL mode is not enabled.) If you want to disable additional engines in this way, you may want to consider building a special binary to be used on the replica that supports only the engines you want.

If you use raw data files (a binary backup) to set up the replica, it is not possible to change the initial table format. Instead, use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to change the table types after the replica has been started.

For new source/replica replication setups where there are currently no tables on the source, avoid specifying the engine type when creating new tables.

If you are already running a replication solution and want to convert your existing tables to another engine type, follow these steps:

Stop the replica from running replication updates:

mysql> **STOP SLAVE;**

Or from MySQL 8.0.22:

mysql> **STOP REPLICA;**

This makes it possible to change engine types without interruption.

Execute an **ALTER TABLE ... ENGINE='*engine\_type*'** for each table to be changed.

Start the replication process again:

mysql> **START SLAVE;**

Or, beginning with MySQL 8.0.22:

mysql> **START REPLICA;**

Although the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) variable is not replicated, be aware that [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements that include the engine specification are replicated to the replica correctly. If, in the case of a [**CSV**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#csv-storage-engine) table, you execute this statement:

mysql> **ALTER TABLE csvtable ENGINE='MyISAM';**

This statement is replicated; the table's engine type on the replica is converted to **InnoDB**, even if you have previously changed the table type on the replica to an engine other than **CSV**. If you want to retain engine differences on the source and replica, you should be careful to use the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) variable on the source when creating a new table. For example, instead of:

mysql> **CREATE TABLE tablea (columna int) Engine=MyISAM;**

Use this format:

mysql> **SET default\_storage\_engine=MyISAM;**

mysql> **CREATE TABLE tablea (columna int);**

When replicated, the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) variable is ignored, and the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement executes on the replica using the replica's default engine.

**17.4.5 Using Replication for Scale-Out**

You can use replication as a scale-out solution; that is, where you want to split up the load of database queries across multiple database servers, within some reasonable limitations.

Because replication works from the distribution of one source to one or more replicas, using replication for scale-out works best in an environment where you have a high number of reads and low number of writes/updates. Most websites fit into this category, where users are browsing the website, reading articles, posts, or viewing products. Updates only occur during session management, or when making a purchase or adding a comment/message to a forum.

Replication in this situation enables you to distribute the reads over the replicas, while still enabling your web servers to communicate with the source when a write is required. You can see a sample replication layout for this scenario in [Figure 17.1, “Using Replication to Improve Performance During Scale-Out”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-scaleout).

**Figure 17.1 Using Replication to Improve Performance During Scale-Out**

If the part of your code that is responsible for database access has been properly abstracted/modularized, converting it to run with a replicated setup should be very smooth and easy. Change the implementation of your database access to send all writes to the source, and to send reads to either the source or a replica. If your code does not have this level of abstraction, setting up a replicated system gives you the opportunity and motivation to clean it up. Start by creating a wrapper library or module that implements the following functions:

**safe\_writer\_connect()**

**safe\_reader\_connect()**

**safe\_reader\_statement()**

**safe\_writer\_statement()**

**safe\_** in each function name means that the function takes care of handling all error conditions. You can use different names for the functions. The important thing is to have a unified interface for connecting for reads, connecting for writes, doing a read, and doing a write.

Then convert your client code to use the wrapper library. This may be a painful and scary process at first, but it pays off in the long run. All applications that use the approach just described are able to take advantage of a source/replica configuration, even one involving multiple replicas. The code is much easier to maintain, and adding troubleshooting options is trivial. You need modify only one or two functions (for example, to log how long each statement took, or which statement among those issued gave you an error).

If you have written a lot of code, you may want to automate the conversion task by writing a conversion script. Ideally, your code uses consistent programming style conventions. If not, then you are probably better off rewriting it anyway, or at least going through and manually regularizing it to use a consistent style.

**17.4.6 Replicating Different Databases to Different Replicas**

There may be situations where you have a single source server and want to replicate different databases to different replicas. For example, you may want to distribute different sales data to different departments to help spread the load during data analysis. A sample of this layout is shown in [Figure 17.2, “Replicating Databases to Separate Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-multi-db).

**Figure 17.2 Replicating Databases to Separate Replicas**

You can achieve this separation by configuring the source and replicas as normal, and then limiting the binary log statements that each replica processes by using the [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) configuration option on each replica.

**Important**

You should *not* use [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) for this purpose when using statement-based replication, since statement-based replication causes this option's effects to vary according to the database that is currently selected. This applies to mixed-format replication as well, since this enables some updates to be replicated using the statement-based format.

However, it should be safe to use [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) for this purpose if you are using row-based replication only, since in this case the currently selected database has no effect on the option's operation.

For example, to support the separation as shown in [Figure 17.2, “Replicating Databases to Separate Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-multi-db), you should configure each replica as follows, before executing [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica):

Replica 1 should use --replicate-wild-do-table=databaseA.%.

Replica 2 should use --replicate-wild-do-table=databaseB.%.

Replica 3 should use --replicate-wild-do-table=databaseC.%.

Each replica in this configuration receives the entire binary log from the source, but executes only those events from the binary log that apply to the databases and tables included by the [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) option in effect on that replica.

If you have data that must be synchronized to the replicas before replication starts, you have a number of choices:

Synchronize all the data to each replica, and delete the databases, tables, or both that you do not want to keep.

Use [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to create a separate dump file for each database and load the appropriate dump file on each replica.

Use a raw data file dump and include only the specific files and databases that you need for each replica.

**Note**

This does not work with [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) databases unless you use [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table).

**17.4.7 Improving Replication Performance**

As the number of replicas connecting to a source increases, the load, although minimal, also increases, as each replica uses a client connection to the source. Also, as each replica must receive a full copy of the source's binary log, the network load on the source may also increase and create a bottleneck.

If you are using a large number of replicas connected to one source, and that source is also busy processing requests (for example, as part of a scale-out solution), then you may want to improve the performance of the replication process.

One way to improve the performance of the replication process is to create a deeper replication structure that enables the source to replicate to only one replica, and for the remaining replicas to connect to this primary replica for their individual replication requirements. A sample of this structure is shown in [Figure 17.3, “Using an Additional Replication Source to Improve Performance”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-performance).

**Figure 17.3 Using an Additional Replication Source to Improve Performance**

For this to work, you must configure the MySQL instances as follows:

Source 1 is the primary source where all changes and updates are written to the database. Binary logging is enabled on both source servers, which is the default.

Source 2 is the replica to the server Source 1 that provides the replication functionality to the remainder of the replicas in the replication structure. Source 2 is the only machine permitted to connect to Source 1. Source 2 has the [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) option enabled (which is the default). With this option, replication instructions from Source 1 are also written to Source 2's binary log so that they can then be replicated to the true replicas.

Replica 1, Replica 2, and Replica 3 act as replicas to Source 2, and replicate the information from Source 2, which actually consists of the upgrades logged on Source 1.

The above solution reduces the client load and the network interface load on the primary source, which should improve the overall performance of the primary source when used as a direct database solution.

If your replicas are having trouble keeping up with the replication process on the source, there are a number of options available:

If possible, put the relay logs and the data files on different physical drives. To do this, set the [**relay\_log**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log) system variable to specify the location of the relay log.

If heavy disk I/O activity for reads of the binary log file and relay log files is an issue, consider increasing the value of the [**rpl\_read\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_read_size) system variable. This system variable controls the minimum amount of data read from the log files, and increasing it might reduce file reads and I/O stalls when the file data is not currently cached by the operating system. Note that a buffer the size of this value is allocated for each thread that reads from the binary log and relay log files, including dump threads on sources and coordinator threads on replicas. Setting a large value might therefore have an impact on memory consumption for servers.

If the replicas are significantly slower than the source, you may want to divide up the responsibility for replicating different databases to different replicas. See [Section 17.4.6, “Replicating Different Databases to Different Replicas”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-partitioning).

If your source makes use of transactions and you are not concerned about transaction support on your replicas, use **MyISAM** or another nontransactional engine on the replicas. See [Section 17.4.4, “Using Replication with Different Source and Replica Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-diffengines).

If your replicas are not acting as sources, and you have a potential solution in place to ensure that you can bring up a source in the event of failure, then you can disable the [**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) system variable on the replicas. This prevents “dumb” replicas from also logging events they have executed into their own binary log.

**17.4.8 Switching Sources During Failover**

You can tell a replica to change to a new source using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). The replica does not check whether the databases on the source are compatible with those on the replica; it simply begins reading and executing events from the specified coordinates in the new source's binary log. In a failover situation, all the servers in the group are typically executing the same events from the same binary log file, so changing the source of the events should not affect the structure or integrity of the database, provided that you exercise care in making the change.

Replicas should be run with binary logging enabled (the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option), which is the default. If you are not using GTIDs for replication, then the replicas should also be run with [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) (logging replica updates is the default). In this way, the replica is ready to become a source without restarting the replica [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld). Assume that you have the structure shown in [Figure 17.4, “Redundancy Using Replication, Initial Structure”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-redundancy-before).

**Figure 17.4 Redundancy Using Replication, Initial Structure**

In this diagram, the **MySQL Source** holds the source database, the **MySQL Replica** hosts are replicas, and the **Web Client** machines are issuing database reads and writes. Web clients that issue only reads (and would normally be connected to the replicas) are not shown, as they do not need to switch to a new server in the event of failure. For a more detailed example of a read/write scale-out replication structure, see [Section 17.4.5, “Using Replication for Scale-Out”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-scaleout).

Each MySQL replica (**Replica 1**, **Replica 2**, and **Replica 3**) is a replica running with binary logging enabled, and with [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates). Because updates received by a replica from the source are not logged in the binary log when [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) is specified, the binary log on each replica is empty initially. If for some reason **MySQL Source** becomes unavailable, you can pick one of the replicas to become the new source. For example, if you pick **Replica 1**, all **Web Clients** should be redirected to **Replica 1**, which writes the updates to its binary log. **Replica 2** and **Replica 3** should then replicate from **Replica 1**.

The reason for running the replica with [--log-slave-updates=OFF](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) is to prevent replicas from receiving updates twice in case you cause one of the replicas to become the new source. If **Replica 1** has [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) enabled, which is the default, it writes any updates that it receives from **Source** in its own binary log. This means that, when **Replica 2** changes from **Source** to **Replica 1** as its source, it may receive updates from **Replica 1** that it has already received from **Source**.

Make sure that all replicas have processed any statements in their relay log. On each replica, issue [**STOP REPLICA | SLAVE IO\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica), then check the output of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) until you see **Has read all relay log**. When this is true for all replicas, they can be reconfigured to the new setup. On the replica **Replica 1** being promoted to become the source, issue [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master).

On the other replicas **Replica 2** and **Replica 3**, use [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) and **CHANGE REPLICATION SOURCE TO SOURCE\_HOST='Replica1'** or **CHANGE MASTER TO MASTER\_HOST='Replica1'** (where **'Replica1'** represents the real host name of **Replica 1**). To use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | **CHANGE MASTER TO**, add all information about how to connect to **Replica 1** from **Replica 2** or **Replica 3** (***user***, ***password***, ***port***). When issuing the statement in this scenario, there is no need to specify the name of the **Replica 1** binary log file or log position to read from, since the first binary log file and position 4, are the defaults. Finally, execute [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) on **Replica 2** and **Replica 3**.

Once the new replication setup is in place, you need to tell each **Web Client** to direct its statements to **Replica 1**. From that point on, all update statements sent by **Web Client** to **Replica 1** are written to the binary log of **Replica 1**, which then contains every update statement sent to **Replica 1** since **Source** stopped.

The resulting server structure is shown in [Figure 17.5, “Redundancy Using Replication, After Source Failure”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#figure_replication-redundancy-after).

**Figure 17.5 Redundancy Using Replication, After Source Failure**

When **Source** becomes available again, you should make it a replica of **Replica 1**. To do this, issue on **Source** the same [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement as that issued on **Replica 2** and **Replica 3** previously. **Source** then becomes a replica of **Replica 1** and picks up the **Web Client** writes that it missed while it was offline.

To make **Source** a source again, use the preceding procedure as if **Replica 1** was unavailable and **Source** was to be the new source. During this procedure, do not forget to run [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) on **Replica 1** before making **Replica 1**, **Replica 2**, and **Replica 3** replicas of **Source**. If you fail to do this, the replicas may pick up stale writes from the **Web Client** applications dating from before the point at which **Source** became unavailable.

You should be aware that there is no synchronization between replicas, even when they share the same source, and thus some replicas might be considerably ahead of others. This means that in some cases the procedure outlined in the previous example might not work as expected. In practice, however, relay logs on all replicas should be relatively close together.

One way to keep applications informed about the location of the source is to have a dynamic DNS entry for the source server. With **bind** you can use nsupdate to update the DNS dynamically.

**17.4.9 Switching Sources with Asynchronous Connection Failover**

Beginning with MySQL 8.0.22, you can use the asynchronous connection failover mechanism to automatically establish an asynchronous (source to replica) replication connection to a new source after the existing connection from a replica to its source fails. The asynchronous connection failover mechanism can be used to keep a replica synchronized with multiple MySQL servers or groups of servers that share data. The list of potential source servers is stored on the replica, and in the event of a connection failure, a new source is selected from the list based on a weighted priority that you set.

From MySQL 8.0.23, the asynchronous connection failover mechanism also supports Group Replication topologies, by automatically monitoring changes to group membership and distinguishing between primary and secondary servers. When you add a group member to the source list and define it as part of a managed group, the asynchronous connection failover mechanism updates the source list to keep it in line with membership changes, adding and removing group members automatically as they join or leave. Only online group members that are in the majority are used for connections and obtaining status. The last remaining member of a managed group is not removed automatically if it leaves the group, so that the configuration of the managed group is kept, but you can delete a managed group manually if it is no longer needed.

To activate asynchronous connection failover for a replication channel set **SOURCE\_CONNECTION\_AUTO\_FAILOVER=1** on the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) for the channel. GTID auto-positioning must be in use for the channel (**SOURCE\_AUTO\_POSITION = 1** | **MASTER\_AUTO\_POSITION = 1**). You can set this option while the replica is running.

**Important**

When the existing connection to a source fails, the replica first retries the same connection the number of times specified by the **SOURCE\_RETRY\_COUNT** | **MASTER\_RETRY\_COUNT** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement. The interval between attempts is set by the **SOURCE\_CONNECT\_RETRY** | **MASTER\_CONNECT\_RETRY** option. When these attempts are exhausted, the asynchronous connection failover mechanism takes over. Note that the defaults for these options, which were designed for a connection to a single source, make the replica retry the same connection for 60 days. To ensure that the asynchronous connection failover mechanism can be activated promptly, set **SOURCE\_RETRY\_COUNT** | **MASTER\_RETRY\_COUNT** and **SOURCE\_CONNECT\_RETRY** | **MASTER\_CONNECT\_RETRY** to minimal numbers that just allow a few retry attempts with the same source, in case the connection failure is caused by a transient network outage. Suitable values are **SOURCE\_RETRY\_COUNT=3** | **MASTER\_RETRY\_COUNT=3** and **SOURCE\_CONNECT\_RETRY=10** | **MASTER\_CONNECT\_RETRY=10**, which make the replica retry the connection 3 times with 10-second intervals between.

Also set a source list on the replica for the replication channel. You set and manage source lists using the [**asynchronous\_connection\_failover\_add\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-source) and [**asynchronous\_connection\_failover\_delete\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-source) UDFs to add and remove single replication source servers. To add and remove managed groups of servers, use the [**asynchronous\_connection\_failover\_add\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-managed) and [**asynchronous\_connection\_failover\_delete\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-managed) UDFs instead.

The UDFs name the relevant replication channel and specify the host name, port number, network namespace, and weighted priority (1-100, with 100 being the highest priority) of a MySQL instance to add to or delete from the channel's source list. For a managed group, you also specify the type of managed service (currently only Group Replication is available), and the identifier of the managed group (for Group Replication, this is the value of the [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) system variable). When you add a managed group, you only need to add one group member, and the replica automatically adds the rest from the current group membership. When you delete a managed group, you delete the entire group together.

In MySQL 8.0.22, the asynchronous connection failover mechanism is activated following the failure of the replica's connection to the source, and it issues a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement to attempt to connect to a new source. In this release, the connection fails over if the replication I/O thread stops due to the source stopping or due to a network failure. The connection does not fail over in any other situations, such as when the replication threads are stopped by a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement.

From MySQL 8.0.23, the asynchronous connection failover mechanism also fails over the connection if another available server on the source list has a higher priority (weight) setting. This function ensures that the replica stays connected to the most suitable source server at all times, and it applies to both managed groups and single (non-managed) servers. For a managed group, a source’s weight is assigned depending on whether it is a primary or a secondary server. So assuming that you set up the managed group to give a higher weight to a primary and a lower weight to a secondary, when the primary changes, the higher weight is assigned to the new primary, so the replica changes over the connection to it. The asynchronous connection failover mechanism additionally changes connection if the currently connected managed source server leaves the managed group, or is no longer in the majority in the managed group.

When failing over a connection, the source with the highest priority (weight) setting among the alternative sources listed in the source list for the channel is chosen for the first connection attempt. . The replica checks first that it can connect to the source server, or in the case of a managed group, that the source server has **ONLINE** status in the group. If the highest weighted source is not available, the replica tries with all the listed sources in descending order of weight, then starts again from the highest weighted source. If multiple sources have the same weight, the replica orders them randomly. If the replica needs to start working through the list again, it includes and retries the source to which the original connection failure occurred.

The source lists are stored in the **mysql.replication\_asynchronous\_connection\_failover** and **mysql.replication\_asynchronous\_connection\_failover\_managed** tables, and can be viewed in the Performance Schema tables [**replication\_asynchronous\_connection\_failover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-asynchronous-connection-failover-table) and [**replication\_asynchronous\_connection\_failover\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-asynchronous-connection-failover-managed-table). The replica uses a monitor thread to track the membership of managed groups and update the source list (**thread/sql/replica\_monitor**). The setting for the **SOURCE\_CONNECTION\_AUTO\_FAILOVER** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, and the source list, are transferred to a clone of the replica during a remote cloning operation.

The requirements for using the asynchronous connection failover mechanism are as follows:

GTIDs must be in use on the source and the replica ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), and the **SOURCE\_AUTO\_POSITION** | **MASTER\_AUTO\_POSITION** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement must be enabled on the replica, so that GTID auto-positioning is used for the connection to the source.

The same replication user account and password must exist on all the source servers in the source list for the channel. This account is used for the connection to each of the sources. You can set up different accounts for different channels.

The replication user account must be given **SELECT** permissions on the Performance Schema tables, for example, by issuing **GRANT SELECT ON performance\_schema.\* TO '*repl\_user*';**

The replication user account and password cannot be specified on the statement used to start replication, because they need to be available on the automatic restart for the connection to the alternative source. They must be set for the channel using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement on the replica, and recorded in the replication metadata repositories.

**17.4.10 Semisynchronous Replication**

[17.4.10.1 Semisynchronous Replication Administrative Interface](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-semisync-interface)

[17.4.10.2 Semisynchronous Replication Installation and Configuration](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-semisync-installation)

[17.4.10.3 Semisynchronous Replication Monitoring](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-semisync-monitoring)

In addition to the built-in asynchronous replication, MySQL 8.0 supports an interface to semisynchronous replication that is implemented by plugins. This section discusses what semisynchronous replication is and how it works. The following sections cover the administrative interface to semisynchronous replication and how to install, configure, and monitor it.

MySQL replication by default is asynchronous. The source writes events to its binary log and replicas request them when they are ready. The source does not know whether or when a replica has retrieved and processed the transactions, and there is no guarantee that any event ever reaches any replica. With asynchronous replication, if the source crashes, transactions that it has committed might not have been transmitted to any replica. Failover from source to replica in this case might result in failover to a server that is missing transactions relative to the source.

With fully synchronous replication, when a source commits a transaction, all replicas have also committed the transaction before the source returns to the session that performed the transaction. Fully synchronous replication means failover from the source to any replica is possible at any time. The drawback of fully synchronous replication is that there might be a lot of delay to complete a transaction.

Semisynchronous replication falls between asynchronous and fully synchronous replication. The source waits until at least one replica has received and logged the events (the required number of replicas is configurable), and then commits the transaction. The source does not wait for all replicas to acknowledge receipt, and it requires only an acknowledgement from the replicas, not that the events have been fully executed and committed on the replica side. Semisynchronous replication therefore guarantees that if the source crashes, all the transactions that it has committed have been transmitted to at least one replica.

Compared to asynchronous replication, semisynchronous replication provides improved data integrity, because when a commit returns successfully, it is known that the data exists in at least two places. Until a semisynchronous source receives acknowledgment from the required number of replicas, the transaction is on hold and not committed.

Compared to fully synchronous replication, semisynchronous replication is faster, because it can be configured to balance your requirements for data integrity (the number of replicas acknowledging receipt of the transaction) with the speed of commits, which are slower due to the need to wait for replicas.

**Important**

With semisynchronous replication, if the source crashes and a failover to a replica is carried out, the failed source should not be reused as the replication source, and should be discarded. It could have transactions that were not acknowledged by any replica, which were therefore not committed before the failover.

If your goal is to implement a fault-tolerant replication topology where all the servers receive the same transactions in the same order, and a server that crashes can rejoin the group and be brought up to date automatically, you can use Group Replication to achieve this. For information, see [Chapter 18, *Group Replication*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html).

The performance impact of semisynchronous replication compared to asynchronous replication is the tradeoff for increased data integrity. The amount of slowdown is at least the TCP/IP roundtrip time to send the commit to the replica and wait for the acknowledgment of receipt by the replica. This means that semisynchronous replication works best for close servers communicating over fast networks, and worst for distant servers communicating over slow networks. Semisynchronous replication also places a rate limit on busy sessions by constraining the speed at which binary log events can be sent from source to replica. When one user is too busy, this slows it down, which can be useful in some deployment situations.

Semisynchronous replication between a source and its replicas operates as follows:

A replica indicates whether it is semisynchronous-capable when it connects to the source.

If semisynchronous replication is enabled on the source side and there is at least one semisynchronous replica, a thread that performs a transaction commit on the source blocks and waits until at least one semisynchronous replica acknowledges that it has received all events for the transaction, or until a timeout occurs.

The replica acknowledges receipt of a transaction's events only after the events have been written to its relay log and flushed to disk.

If a timeout occurs without any replica having acknowledged the transaction, the source reverts to asynchronous replication. When at least one semisynchronous replica catches up, the source returns to semisynchronous replication.

Semisynchronous replication must be enabled on both the source and replica sides. If semisynchronous replication is disabled on the source, or enabled on the source but on no replicas, the source uses asynchronous replication.

While the source is blocking (waiting for acknowledgment from a replica), it does not return to the session that performed the transaction. When the block ends, the source returns to the session, which then can proceed to execute other statements. At this point, the transaction has committed on the source side, and receipt of its events has been acknowledged by at least one replica. The number of replica acknowledgments the source must receive per transaction before returning to the session is configurable using the [**rpl\_semi\_sync\_master\_wait\_for\_slave\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_for_slave_count) system variable, for which the default value is 1.

Blocking also occurs after rollbacks that are written to the binary log, which occurs when a transaction that modifies nontransactional tables is rolled back. The rolled-back transaction is logged even though it has no effect for transactional tables because the modifications to the nontransactional tables cannot be rolled back and must be sent to replicas.

For statements that do not occur in transactional context (that is, when no transaction has been started with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or [**SET autocommit = 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable)), autocommit is enabled and each statement commits implicitly. With semisynchronous replication, the source blocks for each such statement, just as it does for explicit transaction commits.

The [**rpl\_semi\_sync\_master\_wait\_point**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_wait_point) system variable controls the point at which a semisynchronous source server waits for replica acknowledgment of transaction receipt before returning a status to the client that committed the transaction. These values are permitted:

**AFTER\_SYNC** (the default): The source writes each transaction to its binary log and the replica, and syncs the binary log to disk. The source waits for replica acknowledgment of transaction receipt after the sync. Upon receiving acknowledgment, the source commits the transaction to the storage engine and returns a result to the client, which then can proceed.

**AFTER\_COMMIT**: The source writes each transaction to its binary log and the replica, syncs the binary log, and commits the transaction to the storage engine. The source waits for replica acknowledgment of transaction receipt after the commit. Upon receiving acknowledgment, the source returns a result to the client, which then can proceed.

The replication characteristics of these settings differ as follows:

With **AFTER\_SYNC**, all clients see the committed transaction at the same time, which is after it has been acknowledged by the replica and committed to the storage engine on the source. Thus, all clients see the same data on the source.

In the event of source failure, all transactions committed on the source have been replicated to the replica (saved to its relay log). An unexpected exit of the source and failover to the replica is lossless because the replica is up to date. As noted above, the source should not be reused after the failover.

With **AFTER\_COMMIT**, the client issuing the transaction gets a return status only after the server commits to the storage engine and receives replica acknowledgment. After the commit and before replica acknowledgment, other clients can see the committed transaction before the committing client.

If something goes wrong such that the replica does not process the transaction, then in the event of an unexpected source exit and failover to the replica, it is possible for such clients to see a loss of data relative to what they saw on the source.

**17.4.10.1 Semisynchronous Replication Administrative Interface**

The administrative interface to semisynchronous replication has several components:

Two plugins implement semisynchronous capability. There is one plugin for the source side and one for the replica side.

System variables control plugin behavior. Some examples:

[**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled)

Controls whether semisynchronous replication is enabled on the source server. To enable or disable the plugin, set this variable to 1 or 0, respectively. The default is 0 (off).

[**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout)

A value in milliseconds that controls how long the source waits on a commit for acknowledgment from a replica before timing out and reverting to asynchronous replication. The default value is 10000 (10 seconds).

[**rpl\_semi\_sync\_slave\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_enabled)

Similar to [**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled), but controls the replica plugin.

All **rpl\_semi\_sync\_*xxx*** system variables are described at [Section 17.1.6.2, “Replication Source Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-source) and [Section 17.1.6.3, “Replica Server Options and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-options-replica).

From MySQL 8.0.23, you can improve the performance of semisynchronous replication by enabling the system variables [**replication\_sender\_observe\_commit\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_sender_observe_commit_only), which limits callbacks, and [**replication\_optimize\_for\_static\_plugin\_config**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_replication_optimize_for_static_plugin_config), which adds shared locks and avoids unnecessary lock acquisitions. These settings help as the number of replicas increases, because contention for locks can slow down performance. Semisynchronous replication source servers can also get performance benefits from enabling these system variables, because they use the same locking mechanisms as the replicas.

Status variables enable semisynchronous replication monitoring. Some examples:

[**Rpl\_semi\_sync\_master\_clients**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_clients)

The number of semisynchronous replicas.

[**Rpl\_semi\_sync\_master\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_status)

Whether semisynchronous replication currently is operational on the source server. The value is 1 if the plugin has been enabled and a commit acknowledgment has not occurred. It is 0 if the plugin is not enabled or the source has fallen back to asynchronous replication due to commit acknowledgment timeout.

[**Rpl\_semi\_sync\_master\_no\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_no_tx)

The number of commits that were not acknowledged successfully by a replica.

[**Rpl\_semi\_sync\_master\_yes\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_yes_tx)

The number of commits that were acknowledged successfully by a replica.

[**Rpl\_semi\_sync\_slave\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_slave_status)

Whether semisynchronous replication currently is operational on the replica. This is 1 if the plugin has been enabled and the replication I/O thread is running, 0 otherwise.

All **Rpl\_semi\_sync\_*xxx*** status variables are described at [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

The system and status variables are available only if the appropriate source or replica plugin has been installed with [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin).

**17.4.10.2 Semisynchronous Replication Installation and Configuration**

Semisynchronous replication is implemented using plugins, so the plugins must be installed into the server to make them available. After a plugin has been installed, you control it by means of the system variables associated with it. These system variables are unavailable until the associated plugin has been installed.

This section describes how to install the semisynchronous replication plugins. For general information about installing plugins, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

To use semisynchronous replication, the following requirements must be satisfied:

The capability of installing plugins requires a MySQL server that supports dynamic loading. To verify this, check that the value of the [**have\_dynamic\_loading**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_have_dynamic_loading) system variable is **YES**. Binary distributions should support dynamic loading.

Replication must already be working, see [Section 17.1, “Configuring Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-configuration).

There must not be multiple replication channels configured. Semisynchronous replication is only compatible with the default replication channel. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels).

To set up semisynchronous replication, use the following instructions. The [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin), [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica), and [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statements mentioned here require the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

MySQL distributions include semisynchronous replication plugin files for the source side and the replica side.

To be usable by a source or replica server, the appropriate plugin library file must be located in the MySQL plugin directory (the directory named by the [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable). If necessary, configure the plugin directory location by setting the value of [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) at server startup.

The plugin library file base names are **semisync\_master** for the source, and **semisync\_slave** for the replica. The file name suffix differs per platform (for example, .so for Unix and Unix-like systems, .dll for Windows).

The source plugin library file must be present in the plugin directory of the source server. The replica plugin library file must be present in the plugin directory of each replica server.

To load the plugins, use the [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement on the source and on each replica that is to be semisynchronous, adjusting the .so suffix for your platform as necessary.

On the source:

INSTALL PLUGIN rpl\_semi\_sync\_master SONAME 'semisync\_master.so';

On each replica:

INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';

If an attempt to install a plugin results in an error on Linux similar to that shown here, you must install **libimf**:

mysql> **INSTALL PLUGIN rpl\_semi\_sync\_master SONAME 'semisync\_master.so';**

ERROR 1126 (HY000): Can't open shared library

'/usr/local/mysql/lib/plugin/semisync\_master.so'

(errno: 22 libimf.so: cannot open shared object file:

No such file or directory)

You can obtain **libimf** from <https://dev.mysql.com/downloads/os-linux.html>.

To see which plugins are installed, use the [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) statement, or query the [**INFORMATION\_SCHEMA.PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-plugins-table) table.

To verify plugin installation, examine the [**INFORMATION\_SCHEMA.PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-plugins-table) table or use the [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) statement (see [Section 5.6.2, “Obtaining Server Plugin Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#obtaining-plugin-information)). For example:

mysql> **SELECT PLUGIN\_NAME, PLUGIN\_STATUS**

**FROM INFORMATION\_SCHEMA.PLUGINS**

**WHERE PLUGIN\_NAME LIKE '%semi%';**

+----------------------+---------------+

| PLUGIN\_NAME | PLUGIN\_STATUS |

+----------------------+---------------+

| rpl\_semi\_sync\_master | ACTIVE |

+----------------------+---------------+

If the plugin fails to initialize, check the server error log for diagnostic messages.

After a semisynchronous replication plugin has been installed, it is disabled by default. The plugins must be enabled both on the source side and the replica side to enable semisynchronous replication. If only one side is enabled, replication is asynchronous.

To control whether an installed plugin is enabled, set the appropriate system variables. You can set these variables at runtime using [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), or at server startup on the command line or in an option file.

At runtime, these source-side system variables are available:

SET GLOBAL rpl\_semi\_sync\_master\_enabled = {0|1};

SET GLOBAL rpl\_semi\_sync\_master\_timeout = ***N***;

On the replica side, this system variable is available:

SET GLOBAL rpl\_semi\_sync\_slave\_enabled = {0|1};

For [**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled) or [**rpl\_semi\_sync\_slave\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_slave_enabled), the value should be 1 to enable semisynchronous replication or 0 to disable it. By default, these variables are set to 0.

For [**rpl\_semi\_sync\_master\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_timeout), the value ***N*** is given in milliseconds. The default value is 10000 (10 seconds).

If you enable semisynchronous replication on a replica at runtime, you must also start the replication I/O thread (stopping it first if it is already running) to cause the replica to connect to the source and register as a semisynchronous replica:

STOP SLAVE IO\_THREAD;

START SLAVE IO\_THREAD;

Or from MySQL 8.0.22:

STOP REPLICA IO\_THREAD;

START REPLICA IO\_THREAD;

If the replication I/O thread is already running and you do not restart it, the replica continues to use asynchronous replication.

At server startup, the variables that control semisynchronous replication can be set as command-line options or in an option file. A setting listed in an option file takes effect each time the server starts. For example, you can set the variables in my.cnf files on the source and replica servers as follows.

On the source:

[mysqld]

rpl\_semi\_sync\_master\_enabled=1

rpl\_semi\_sync\_master\_timeout=1000 # 1 second

On each replica:

[mysqld]

rpl\_semi\_sync\_slave\_enabled=1

**17.4.10.3 Semisynchronous Replication Monitoring**

The plugins for the semisynchronous replication capability expose several system and status variables that you can examine to determine its configuration and operational state.

The system variable reflect how semisynchronous replication is configured. To check their values, use [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables):

mysql> **SHOW VARIABLES LIKE 'rpl\_semi\_sync%';**

The status variables enable you to monitor the operation of semisynchronous replication. To check their values, use [**SHOW STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status):

mysql> **SHOW STATUS LIKE 'Rpl\_semi\_sync%';**

When the source switches between asynchronous or semisynchronous replication due to commit-blocking timeout or a replica catching up, it sets the value of the [**Rpl\_semi\_sync\_master\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_status) status variable appropriately. Automatic fallback from semisynchronous to asynchronous replication on the source means that it is possible for the [**rpl\_semi\_sync\_master\_enabled**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_semi_sync_master_enabled) system variable to have a value of 1 on the source side even when semisynchronous replication is in fact not operational at the moment. You can monitor the [**Rpl\_semi\_sync\_master\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_status) status variable to determine whether the source currently is using asynchronous or semisynchronous replication.

To see how many semisynchronous replicas are connected, check [**Rpl\_semi\_sync\_master\_clients**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_clients).

The number of commits that have been acknowledged successfully or unsuccessfully by replicas are indicated by the [**Rpl\_semi\_sync\_master\_yes\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_yes_tx) and [**Rpl\_semi\_sync\_master\_no\_tx**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_master_no_tx) variables.

On the replica side, [**Rpl\_semi\_sync\_slave\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Rpl_semi_sync_slave_status) indicates whether semisynchronous replication currently is operational.

**17.4.11 Delayed Replication**

MySQL supports delayed replication such that a replica server deliberately executes transactions later than the source by at least a specified amount of time. This section describes how to configure a replication delay on a replica, and how to monitor replication delay.

In MySQL 8.0, the method of delaying replication depends on two timestamps, **immediate\_commit\_timestamp** and **original\_commit\_timestamp** (see [Replication Delay Timestamps](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-delayed-timestamps)). If all servers in the replication topology are running MySQL 8.0 or above, delayed replication is measured using these timestamps. If either the immediate source or replica is not using these timestamps, the implementation of delayed replication from MySQL 5.7 is used (see [Delayed Replication](https://dev.mysql.com/doc/refman/5.7/en/replication-delayed.html)). This section describes delayed replication between servers which are all using these timestamps.

The default replication delay is 0 seconds. Use a [**CHANGE REPLICATION SOURCE TO SOURCE\_DELAY=*N***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or a **CHANGE MASTER TO MASTER\_DELAY=*N*** statement (before MySQL 8.0.23) to set the delay to ***N*** seconds. A transaction received from the source is not executed until at least ***N*** seconds later than its commit on the immediate source. The delay happens per transaction (not event as in previous MySQL versions) and the actual delay is imposed only on **gtid\_log\_event** or **anonymous\_gtid\_log\_event**. The other events in the transaction always follow these events without any waiting time imposed on them.

**Note**

[**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) and [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) take effect immediately and ignore any delay. [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the delay to 0.

The [**replication\_applier\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-configuration-table) Performance Schema table contains the **DESIRED\_DELAY** column which shows the delay configured using the **SOURCE\_DELAY** | **MASTER\_DELAY** option. The [**replication\_applier\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-table) Performance Schema table contains the **REMAINING\_DELAY** column which shows the number of delay seconds remaining.

Delayed replication can be used for several purposes:

To protect against user mistakes on the source. With a delay you can roll back a delayed replica to the time just before the mistake.

To test how the system behaves when there is a lag. For example, in an application, a lag might be caused by a heavy load on the replica. However, it can be difficult to generate this load level. Delayed replication can simulate the lag without having to simulate the load. It can also be used to debug conditions related to a lagging replica.

To inspect what the database looked like in the past, without having to reload a backup. For example, by configuring a replica with a delay of one week, if you then need to see what the database looked like before the last few days' worth of development, the delayed replica can be inspected.

**Replication Delay Timestamps**

MySQL 8.0 provides a new method for measuring delay (also referred to as replication lag) in replication topologies that depends on the following timestamps associated with the GTID of each transaction (instead of each event) written to the binary log.

**original\_commit\_timestamp**: the number of microseconds since epoch when the transaction was written (committed) to the binary log of the original source.

**immediate\_commit\_timestamp**: the number of microseconds since epoch when the transaction was written (committed) to the binary log of the immediate source.

The output of [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) displays these timestamps in two formats, microseconds from epoch and also **TIMESTAMP** format, which is based on the user defined time zone for better readability. For example:

#170404 10:48:05 server id 1 end\_log\_pos 233 CRC32 0x016ce647 GTID last\_committed=0

\ sequence\_number=1 original\_committed\_timestamp=1491299285661130 immediate\_commit\_timestamp=1491299285843771

# original\_commit\_timestamp=1491299285661130 (2017-04-04 10:48:05.661130 WEST)

# immediate\_commit\_timestamp=1491299285843771 (2017-04-04 10:48:05.843771 WEST)

/\*!80001 SET @@SESSION.original\_commit\_timestamp=1491299285661130\*//\*!\*/;

SET @@SESSION.GTID\_NEXT= 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1'/\*!\*/;

# at 233

As a rule, the **original\_commit\_timestamp** is always the same on all replicas where the transaction is applied. In source-replica replication, the **original\_commit\_timestamp** of a transaction in the (original) source’s binary log is always the same as its **immediate\_commit\_timestamp**. In the replica’s relay log, the **original\_commit\_timestamp** and **immediate\_commit\_timestamp** of the transaction are the same as in the source’s binary log; whereas in its own binary log, the transaction’s **immediate\_commit\_timestamp** corresponds to when the replica committed the transaction.

In a Group Replication setup, when the original source is a member of a group, the **original\_commit\_timestamp** is generated when the transaction is ready to be committed. In other words, when it finished executing on the original source and its write set is ready to be sent to all members of the group for certification. When the original source is a server outside the group, the **original\_commit\_timestamp** is preserved. The same **original\_commit\_timestamp** for a particular transaction is replicated to all servers in the group, and to any replica outside the group that is replicating from a member. From MySQL 8.0.25, each recipient of the transaction also stores the local commit time in its binary log using **immediate\_commit\_timestamp**.

View change events, which are exclusive to Group Replication, are a special case. Transactions containing these events are generated by each group member but share the same GTID (so, they are not first executed in a source and then replicated to the group, but all members of the group execute and apply the same transaction). Before MySQL 8.0.25, these transactions have their **original\_commit\_timestamp** set to zero, and they appear this way in viewable output. From MySQL 8.0.25, for improved observability, group members set local timestamp values for transactions associated with view change events.

**Monitoring Replication Delay**

One of the most common ways to monitor replication delay (lag) in previous MySQL versions was by relying on the **Seconds\_Behind\_Master** field in the output of **SHOW REPLICA | SLAVE STATUS**. However, this metric is not suitable when using replication topologies more complex than the traditional source-replica setup, such as Group Replication. The addition of **immediate\_commit\_timestamp** and **original\_commit\_timestamp** to MySQL 8 provides a much finer degree of information about replication delay. The recommended method to monitor replication delay in a topology that supports these timestamps is using the following Performance Schema tables.

[**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table): current status of the connection to the source, provides information on the last and current transaction the connection thread queued into the relay log.

[**replication\_applier\_status\_by\_coordinator**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-coordinator-table): current status of the coordinator thread that only displays information when using a multithreaded replica, provides information on the last transaction buffered by the coordinator thread to a worker’s queue, as well as the transaction it is currently buffering.

[**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table): current status of the thread(s) applying transactions received from the source, provides information about the transactions applied by the replication SQL thread, or by each worker thread when using a multithreaded replica.

Using these tables you can monitor information about the last transaction the corresponding thread processed and the transaction that thread is currently processing. This information comprises:

a transaction’s GTID

a transaction's **original\_commit\_timestamp** and **immediate\_commit\_timestamp**, retrieved from the replica’s relay log

the time a thread started processing a transaction

for the last processed transaction, the time the thread finished processing it

In addition to the Performance Schema tables, the output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) has three fields that show:

**SQL\_Delay**: A nonnegative integer indicating the replication delay configured using [**CHANGE REPLICATION SOURCE TO SOURCE\_DELAY=*N***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) (from MySQL 8.0.23) or **CHANGE MASTER TO MASTER\_DELAY=N** (before MySQL 8.0.23), measured in seconds.

**SQL\_Remaining\_Delay**: When **Replica\_SQL\_Running\_State** is **Waiting until MASTER\_DELAY seconds after master executed event**, this field contains an integer indicating the number of seconds left of the delay. At other times, this field is **NULL**.

**Replica\_SQL\_Running\_State**: A string indicating the state of the SQL thread (analogous to **Replica\_IO\_State**). The value is identical to the **State** value of the SQL thread as displayed by [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist).

When the replication SQL thread is waiting for the delay to elapse before executing an event, [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) displays its **State** value as **Waiting until MASTER\_DELAY seconds after master executed event**.

**17.5 Replication Notes and Tips**

[17.5.1 Replication Features and Issues](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features)

[17.5.2 Replication Compatibility Between MySQL Versions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-compatibility)

[17.5.3 Upgrading a Replication Setup](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-upgrade)

[17.5.4 Troubleshooting Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-problems)

[17.5.5 How to Report Replication Bugs or Problems](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-bugs)

**17.5.1 Replication Features and Issues**

[17.5.1.1 Replication and AUTO\_INCREMENT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-auto-increment)

[17.5.1.2 Replication and BLACKHOLE Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-blackhole)

[17.5.1.3 Replication and Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-charset)

[17.5.1.4 Replication and CHECKSUM TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-checksum-table)

[17.5.1.5 Replication of CREATE SERVER, ALTER SERVER, and DROP SERVER](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-create-alter-drop-server)

[17.5.1.6 Replication of CREATE ... IF NOT EXISTS Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-create-if-not-exists)

[17.5.1.7 Replication of CREATE TABLE ... SELECT Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-create-select)

[17.5.1.8 Replication of CURRENT\_USER()](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-current-user)

[17.5.1.9 Replication with Differing Table Definitions on Source and Replica](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-differing-tables)

[17.5.1.10 Replication and DIRECTORY Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-directory)

[17.5.1.11 Replication of DROP ... IF EXISTS Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-drop-if-exists)

[17.5.1.12 Replication and Floating-Point Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-floatvalues)

[17.5.1.13 Replication and FLUSH](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-flush)

[17.5.1.14 Replication and System Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-functions)

[17.5.1.15 Replication and Fractional Seconds Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-fractional-seconds)

[17.5.1.16 Replication of Invoked Features](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked)

[17.5.1.17 Replication of JSON Documents](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-json)

[17.5.1.18 Replication and LIMIT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-limit)

[17.5.1.19 Replication and LOAD DATA](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-load-data)

[17.5.1.20 Replication and max\_allowed\_packet](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-max-allowed-packet)

[17.5.1.21 Replication and MEMORY Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-memory)

[17.5.1.22 Replication of the mysql System Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-mysqldb)

[17.5.1.23 Replication and the Query Optimizer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-optimizer)

[17.5.1.24 Replication and Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-partitioning)

[17.5.1.25 Replication and REPAIR TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-repair-table)

[17.5.1.26 Replication and Reserved Words](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-reserved-words)

[17.5.1.27 Replication and Row Searches](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-row-searches)

[17.5.1.28 Replication and Source or Replica Shutdowns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-shutdowns)

[17.5.1.29 Replica Errors During Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-slaveerrors)

[17.5.1.30 Replication and Server SQL Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-sql-mode)

[17.5.1.31 Replication and Temporary Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-temptables)

[17.5.1.32 Replication Retries and Timeouts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-timeout)

[17.5.1.33 Replication and Time Zones](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-timezone)

[17.5.1.34 Replication and Transaction Inconsistencies](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies)

[17.5.1.35 Replication and Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transactions)

[17.5.1.36 Replication and Triggers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-triggers)

[17.5.1.37 Replication and TRUNCATE TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-truncate)

[17.5.1.38 Replication and User Name Length](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-user-names)

[17.5.1.39 Replication and Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-variables)

[17.5.1.40 Replication and Views](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-views)

The following sections provide information about what is supported and what is not in MySQL replication, and about specific issues and situations that may occur when replicating certain statements.

Statement-based replication depends on compatibility at the SQL level between the source and replica. In other words, successful statement-based replication requires that any SQL features used be supported by both the source and the replica servers. If you use a feature on the source server that is available only in the current version of MySQL, you cannot replicate to a replica that uses an earlier version of MySQL. Such incompatibilities can also occur within a release series as well as between versions.

If you are planning to use statement-based replication between MySQL 8.0 and a previous MySQL release series, it is a good idea to consult the edition of the *MySQL Reference Manual* corresponding to the earlier release series for information regarding the replication characteristics of that series.

With MySQL's statement-based replication, there may be issues with replicating stored routines or triggers. You can avoid these issues by using MySQL's row-based replication instead. For a detailed list of issues, see [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging). For more information about row-based logging and row-based replication, see [Section 5.4.4.1, “Binary Logging Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-formats), and [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).

For additional information specific to replication and **InnoDB**, see [Section 15.19, “InnoDB and MySQL Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-and-mysql-replication). For information relating to replication with NDB Cluster, see [Section 23.6, “NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication).

**17.5.1.1 Replication and AUTO\_INCREMENT**

Statement-based replication of **AUTO\_INCREMENT**, [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id), and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values is carried out subject to the following exceptions:

A statement invoking a trigger or function that causes an update to an **AUTO\_INCREMENT** column is not replicated correctly using statement-based replication. These statements are marked as unsafe. (Bug #45677)

An [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) into a table that has a composite primary key that includes an **AUTO\_INCREMENT** column that is not the first column of this composite key is not safe for statement-based logging or replication. These statements are marked as unsafe. (Bug #11754117, Bug #45670)

This issue does not affect tables using the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine, since an **InnoDB** table with an [AUTO\_INCREMENT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_auto_increment) column requires at least one key where the auto-increment column is the only or leftmost column.

Adding an **AUTO\_INCREMENT** column to a table with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) might not produce the same ordering of the rows on the replica and the source. This occurs because the order in which the rows are numbered depends on the specific storage engine used for the table and the order in which the rows were inserted. If it is important to have the same order on the source and replica, the rows must be ordered before assigning an **AUTO\_INCREMENT** number. Assuming that you want to add an **AUTO\_INCREMENT** column to a table **t1** that has columns **col1** and **col2**, the following statements produce a new table **t2** identical to **t1** but with an **AUTO\_INCREMENT** column:

CREATE TABLE t2 LIKE t1;

ALTER TABLE t2 ADD id INT AUTO\_INCREMENT PRIMARY KEY;

INSERT INTO t2 SELECT \* FROM t1 ORDER BY col1, col2;

**Important**

To guarantee the same ordering on both source and replica, the **ORDER BY** clause must name *all* columns of **t1**.

The instructions just given are subject to the limitations of [**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like): Foreign key definitions are ignored, as are the **DATA DIRECTORY** and **INDEX DIRECTORY** table options. If a table definition includes any of those characteristics, create **t2** using a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement that is identical to the one used to create **t1**, but with the addition of the **AUTO\_INCREMENT** column.

Regardless of the method used to create and populate the copy having the **AUTO\_INCREMENT** column, the final step is to drop the original table and then rename the copy:

DROP t1;

ALTER TABLE t2 RENAME t1;

See also [Section B.3.6.1, “Problems with ALTER TABLE”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#alter-table-problems).

**17.5.1.2 Replication and BLACKHOLE Tables**

The [**BLACKHOLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#blackhole-storage-engine) storage engine accepts data but discards it and does not store it. When performing binary logging, all inserts to such tables are always logged, regardless of the logging format in use. Updates and deletes are handled differently depending on whether statement based or row based logging is in use. With the statement based logging format, all statements affecting **BLACKHOLE** tables are logged, but their effects ignored. When using row-based logging, updates and deletes to such tables are simply skipped—they are not written to the binary log. A warning is logged whenever this occurs.

For this reason we recommend when you replicate to tables using the [**BLACKHOLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#blackhole-storage-engine) storage engine that you have the [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) server variable set to **STATEMENT**, and not to either **ROW** or **MIXED**.

**17.5.1.3 Replication and Character Sets**

The following applies to replication between MySQL servers that use different character sets:

If the source has databases with a character set different from the global [**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server) value, you should design your [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements so that they do not implicitly rely on the database default character set. A good workaround is to state the character set and collation explicitly in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements.

**17.5.1.4 Replication and CHECKSUM TABLE**

**[CHECKSUM TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "checksum-table" \o "13.7.3.3 CHECKSUM TABLE Statement)** returns a checksum that is calculated row by row, using a method that depends on the table row storage format. The storage format is not guaranteed to remain the same between MySQL versions, so the checksum value might change following an upgrade.

**17.5.1.5 Replication of CREATE SERVER, ALTER SERVER, and DROP SERVER**

The statements [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server), [**ALTER SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-server), and [**DROP SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-server) are not written to the binary log, regardless of the binary logging format that is in use.

**17.5.1.6 Replication of CREATE ... IF NOT EXISTS Statements**

MySQL applies these rules when various **CREATE ... IF NOT EXISTS** statements are replicated:

Every [**CREATE DATABASE IF NOT EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement is replicated, whether or not the database already exists on the source.

Similarly, every [**CREATE TABLE IF NOT EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement without a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is replicated, whether or not the table already exists on the source. This includes [**CREATE TABLE IF NOT EXISTS ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like). Replication of [**CREATE TABLE IF NOT EXISTS ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) follows somewhat different rules; see [Section 17.5.1.7, “Replication of CREATE TABLE ... SELECT Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-create-select), for more information.

[**CREATE EVENT IF NOT EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) is always replicated, whether or not the event named in the statement already exists on the source.

**17.5.1.7 Replication of CREATE TABLE ... SELECT Statements**

MySQL applies these rules when [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements are replicated:

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) always performs an implicit commit ([Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit)).

If the destination table does not exist, logging occurs as follows. It does not matter whether **IF NOT EXISTS** is present.

**STATEMENT** or **MIXED** format: The statement is logged as written.

**ROW** format: The statement is logged as a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement followed by a series of insert-row events.

Prior to MySQL 8.0.21, the statement is logged as two transactions. As of MySQL 8.0.21, on storage engines that support atomic DDL, it is logged as one transaction. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

If the [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement fails, nothing is logged. This includes the case that the destination table exists and **IF NOT EXISTS** is not given.

If the destination table exists and **IF NOT EXISTS** is given, MySQL 8.0 ignores the statement completely; nothing is inserted or logged.

MySQL 8.0 does not allow a [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement to make any changes in tables other than the table that is created by the statement.

**17.5.1.8 Replication of CURRENT\_USER()**

The following statements support use of the [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function to take the place of the name of, and possibly the host for, an affected user or a definer:

[**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user)

[**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user)

[**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant)

[**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke)

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function)

[**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure)

[**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger)

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event)

[**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view)

[**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event)

[**ALTER VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-view)

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password)

When binary logging is enabled and [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) or [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) is used as the definer in any of these statements, MySQL Server ensures that the statement is applied to the same user on both the source and the replica when the statement is replicated. In some cases, such as statements that change passwords, the function reference is expanded before it is written to the binary log, so that the statement includes the user name. For all other cases, the name of the current user on the source is replicated to the replica as metadata, and the replica applies the statement to the current user named in the metadata, rather than to the current user on the replica.

**17.5.1.9 Replication with Differing Table Definitions on Source and Replica**

Source and target tables for replication do not have to be identical. A table on the source can have more or fewer columns than the replica's copy of the table. In addition, corresponding table columns on the source and the replica can use different data types, subject to certain conditions.

**Note**

Replication between tables which are partitioned differently from one another is not supported. See [Section 17.5.1.24, “Replication and Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-partitioning).

In all cases where the source and target tables do not have identical definitions, the database and table names must be the same on both the source and the replica. Additional conditions are discussed, with examples, in the following two sections.

**17.5.1.9.1 Replication with More Columns on Source or Replica**

You can replicate a table from the source to the replica such that the source and replica copies of the table have differing numbers of columns, subject to the following conditions:

Columns common to both versions of the table must be defined in the same order on the source and the replica. (This is true even if both tables have the same number of columns.)

Columns common to both versions of the table must be defined before any additional columns.

This means that executing an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement on the replica where a new column is inserted into the table within the range of columns common to both tables causes replication to fail, as shown in the following example:

Suppose that a table **t**, existing on the source and the replica, is defined by the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

CREATE TABLE t (

c1 INT,

c2 INT,

c3 INT

);

Suppose that the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement shown here is executed on the replica:

ALTER TABLE t ADD COLUMN cnew1 INT AFTER c3;

The previous [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) is permitted on the replica because the columns **c1**, **c2**, and **c3** that are common to both versions of table **t** remain grouped together in both versions of the table, before any columns that differ.

However, the following [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement cannot be executed on the replica without causing replication to break:

ALTER TABLE t ADD COLUMN cnew2 INT AFTER c2;

Replication fails after execution on the replica of the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement just shown, because the new column **cnew2** comes between columns common to both versions of **t**.

Each “extra” column in the version of the table having more columns must have a default value.

A column's default value is determined by a number of factors, including its type, whether it is defined with a **DEFAULT** option, whether it is declared as **NULL**, and the server SQL mode in effect at the time of its creation; for more information, see [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults)).

In addition, when the replica's copy of the table has more columns than the source's copy, each column common to the tables must use the same data type in both tables.

**Examples.** The following examples illustrate some valid and invalid table definitions:

**More columns on the source.** The following table definitions are valid and replicate correctly:

source> **CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);**

replica> **CREATE TABLE t1 (c1 INT, c2 INT);**

The following table definitions would raise an error because the definitions of the columns common to both versions of the table are in a different order on the replica than they are on the source:

source> **CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);**

replica> **CREATE TABLE t1 (c2 INT, c1 INT);**

The following table definitions would also raise an error because the definition of the extra column on the source appears before the definitions of the columns common to both versions of the table:

source> **CREATE TABLE t1 (c3 INT, c1 INT, c2 INT);**

replica> **CREATE TABLE t1 (c1 INT, c2 INT);**

**More columns on the replica.** The following table definitions are valid and replicate correctly:

source> **CREATE TABLE t1 (c1 INT, c2 INT);**

replica> **CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);**

The following definitions raise an error because the columns common to both versions of the table are not defined in the same order on both the source and the replica:

source> **CREATE TABLE t1 (c1 INT, c2 INT);**

replica> **CREATE TABLE t1 (c2 INT, c1 INT, c3 INT);**

The following table definitions also raise an error because the definition for the extra column in the replica's version of the table appears before the definitions for the columns which are common to both versions of the table:

source> **CREATE TABLE t1 (c1 INT, c2 INT);**

replica> **CREATE TABLE t1 (c3 INT, c1 INT, c2 INT);**

The following table definitions fail because the replica's version of the table has additional columns compared to the source's version, and the two versions of the table use different data types for the common column **c2**:

source> **CREATE TABLE t1 (c1 INT, c2 BIGINT);**

replica> **CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);**

**17.5.1.9.2 Replication of Columns Having Different Data Types**

Corresponding columns on the source's and the replica's copies of the same table ideally should have the same data type. However, this is not always strictly enforced, as long as certain conditions are met.

It is usually possible to replicate from a column of a given data type to another column of the same type and same size or width, where applicable, or larger. For example, you can replicate from a **CHAR(10)** column to another **CHAR(10)**, or from a **CHAR(10)** column to a **CHAR(25)** column without any problems. In certain cases, it also possible to replicate from a column having one data type (on the source) to a column having a different data type (on the replica); when the data type of the source's version of the column is promoted to a type that is the same size or larger on the replica, this is known as attribute promotion.

Attribute promotion can be used with both statement-based and row-based replication, and is not dependent on the storage engine used by either the source or the replica. However, the choice of logging format does have an effect on the type conversions that are permitted; the particulars are discussed later in this section.

**Important**

Whether you use statement-based or row-based replication, the replica's copy of the table cannot contain more columns than the source's copy if you wish to employ attribute promotion.

**Statement-based replication.** When using statement-based replication, a simple rule of thumb to follow is, “If the statement run on the source would also execute successfully on the replica, it should also replicate successfully”. In other words, if the statement uses a value that is compatible with the type of a given column on the replica, the statement can be replicated. For example, you can insert any value that fits in a **TINYINT** column into a **BIGINT** column as well; it follows that, even if you change the type of a **TINYINT** column in the replica's copy of a table to **BIGINT**, any insert into that column on the source that succeeds should also succeed on the replica, since it is impossible to have a legal **TINYINT** value that is large enough to exceed a **BIGINT** column.

**Row-based replication: attribute promotion and demotion.** Row-based replication supports attribute promotion and demotion between smaller data types and larger types. It is also possible to specify whether or not to permit lossy (truncated) or non-lossy conversions of demoted column values, as explained later in this section.

**Lossy and non-lossy conversions.** In the event that the target type cannot represent the value being inserted, a decision must be made on how to handle the conversion. If we permit the conversion but truncate (or otherwise modify) the source value to achieve a “fit” in the target column, we make what is known as a lossy conversion. A conversion which does not require truncation or similar modifications to fit the source column value in the target column is a non-lossy conversion.

**Type conversion modes.** The setting of the **slave\_type\_conversions** global server variable controls the type conversion mode used on the replica. This variable takes a set of values from the following list, which describes the effects of each mode on the replica's type-conversion behavior:

**ALL\_LOSSY**

In this mode, type conversions that would mean loss of information are permitted.

This does not imply that non-lossy conversions are permitted, merely that only cases requiring either lossy conversions or no conversion at all are permitted; for example, enabling *only* this mode permits an **INT** column to be converted to **TINYINT** (a lossy conversion), but not a **TINYINT** column to an **INT** column (non-lossy). Attempting the latter conversion in this case would cause replication to stop with an error on the replica.

**ALL\_NON\_LOSSY**

This mode permits conversions that do not require truncation or other special handling of the source value; that is, it permits conversions where the target type has a wider range than the source type.

Setting this mode has no bearing on whether lossy conversions are permitted; this is controlled with the **ALL\_LOSSY** mode. If only **ALL\_NON\_LOSSY** is set, but not **ALL\_LOSSY**, then attempting a conversion that would result in the loss of data (such as **INT** to **TINYINT**, or **CHAR(25)** to **VARCHAR(20)**) causes the replica to stop with an error.

**ALL\_LOSSY,ALL\_NON\_LOSSY**

When this mode is set, all supported type conversions are permitted, whether or not they are lossy conversions.

**ALL\_SIGNED**

Treat promoted integer types as signed values (the default behavior).

**ALL\_UNSIGNED**

Treat promoted integer types as unsigned values.

**ALL\_SIGNED,ALL\_UNSIGNED**

Treat promoted integer types as signed if possible, otherwise as unsigned.

**[***empty***]**

When **slave\_type\_conversions** is not set, no attribute promotion or demotion is permitted; this means that all columns in the source and target tables must be of the same types.

This mode is the default.

When an integer type is promoted, its signedness is not preserved. By default, the replica treats all such values as signed. You can control this behavior using **ALL\_SIGNED**, **ALL\_UNSIGNED**, or both. **ALL\_SIGNED** tells the replica to treat all promoted integer types as signed; **ALL\_UNSIGNED** instructs it to treat these as unsigned. Specifying both causes the replica to treat the value as signed if possible, otherwise to treat it as unsigned; the order in which they are listed is not significant. Neither **ALL\_SIGNED** nor **ALL\_UNSIGNED** has any effect if at least one of **ALL\_LOSSY** or **ALL\_NONLOSSY** is not also used.

Changing the type conversion mode requires restarting the replica with the new **slave\_type\_conversions** setting.

**Supported conversions.** Supported conversions between different but similar data types are shown in the following list:

Between any of the integer types [**TINYINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**SMALLINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**MEDIUMINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), and [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

This includes conversions between the signed and unsigned versions of these types.

Lossy conversions are made by truncating the source value to the maximum (or minimum) permitted by the target column. For ensuring non-lossy conversions when going from unsigned to signed types, the target column must be large enough to accommodate the range of values in the source column. For example, you can demote **TINYINT UNSIGNED** non-lossily to **SMALLINT**, but not to **TINYINT**.

Between any of the decimal types [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types), [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types), and [**NUMERIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types).

**FLOAT** to **DOUBLE** is a non-lossy conversion; **DOUBLE** to **FLOAT** can only be handled lossily. A conversion from **DECIMAL(*M*,*D*)** to **DECIMAL(*M'*,*D'*)** where ***D'* >= *D*** and **(*M'*-*D'*) >= (*M*-*D***) is non-lossy; for any case where ***M'* < *M***, ***D'* < *D***, or both, only a lossy conversion can be made.

For any of the decimal types, if a value to be stored cannot be fit in the target type, the value is rounded down according to the rounding rules defined for the server elsewhere in the documentation. See [Section 12.25.4, “Rounding Behavior”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#precision-math-rounding), for information about how this is done for decimal types.

Between any of the string types [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), including conversions between different widths.

Conversion of a **CHAR**, **VARCHAR**, or **TEXT** to a **CHAR**, **VARCHAR**, or **TEXT** column the same size or larger is never lossy. Lossy conversion is handled by inserting only the first ***N*** characters of the string on the replica, where ***N*** is the width of the target column.

**Important**

Replication between columns using different character sets is not supported.

Between any of the binary data types [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), including conversions between different widths.

Conversion of a **BINARY**, **VARBINARY**, or **BLOB** to a **BINARY**, **VARBINARY**, or **BLOB** column the same size or larger is never lossy. Lossy conversion is handled by inserting only the first ***N*** bytes of the string on the replica, where ***N*** is the width of the target column.

Between any 2 [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type) columns of any 2 sizes.

When inserting a value from a **BIT(*M*)** column into a **BIT(*M'*)** column, where ***M'* > *M***, the most significant bits of the **BIT(*M'*)** columns are cleared (set to zero) and the ***M*** bits of the **BIT(*M*)** value are set as the least significant bits of the **BIT(*M'*)** column.

When inserting a value from a source **BIT(*M*)** column into a target **BIT(*M'*)** column, where ***M'* < *M***, the maximum possible value for the **BIT(*M'*)** column is assigned; in other words, an “all-set” value is assigned to the target column.

Conversions between types not in the previous list are not permitted.

**17.5.1.10 Replication and DIRECTORY Table Options**

If a **DATA DIRECTORY** or **INDEX DIRECTORY** table option is used in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement on the source server, the table option is also used on the replica. This can cause problems if no corresponding directory exists in the replica host file system or if it exists but is not accessible to the replica MySQL server. This can be overridden by using the [**NO\_DIR\_IN\_CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_dir_in_create) server SQL mode on the replica, which causes the replica to ignore the **DATA DIRECTORY** and **INDEX DIRECTORY** table options when replicating [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements. The result is that **MyISAM** data and index files are created in the table's database directory.

For more information, see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

**17.5.1.11 Replication of DROP ... IF EXISTS Statements**

The [**DROP DATABASE IF EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), [**DROP TABLE IF EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), and [**DROP VIEW IF EXISTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) statements are always replicated, even if the database, table, or view to be dropped does not exist on the source. This is to ensure that the object to be dropped no longer exists on either the source or the replica, once the replica has caught up with the source.

**DROP ... IF EXISTS** statements for stored programs (stored procedures and functions, triggers, and events) are also replicated, even if the stored program to be dropped does not exist on the source.

**17.5.1.12 Replication and Floating-Point Values**

With statement-based replication, values are converted from decimal to binary. Because conversions between decimal and binary representations of them may be approximate, comparisons involving floating-point values are inexact. This is true for operations that use floating-point values explicitly, or that use values that are converted to floating-point implicitly. Comparisons of floating-point values might yield different results on source and replica servers due to differences in computer architecture, the compiler used to build MySQL, and so forth. See [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion), and [Section B.3.4.8, “Problems with Floating-Point Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#problems-with-float).

**17.5.1.13 Replication and FLUSH**

Some forms of the [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement are not logged because they could cause problems if replicated to a replica: [**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs) and [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock). For a syntax example, see [Section 13.7.8.3, “FLUSH Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush). The [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables), [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), and [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statements are written to the binary log and thus replicated to replicas. This is not normally a problem because these statements do not modify table data.

However, this behavior can cause difficulties under certain circumstances. If you replicate the privilege tables in the **mysql** database and update those tables directly without using [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), you must issue a [**FLUSH PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-privileges) on the replicas to put the new privileges into effect. In addition, if you use [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) when renaming a **MyISAM** table that is part of a **MERGE** table, you must issue [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) manually on the replicas. These statements are written to the binary log unless you specify **NO\_WRITE\_TO\_BINLOG** or its alias **LOCAL**.

**17.5.1.14 Replication and System Functions**

Certain functions do not replicate well under some conditions:

The [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user), [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) (or [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user)), [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid), [**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version), and [**LOAD\_FILE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file) functions are replicated without change and thus do not work reliably on the replica unless row-based replication is enabled. (See [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).)

[**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) and [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) are automatically replicated using row-based replication when using **MIXED** mode, and generate a warning in **STATEMENT** mode. (See also [Section 17.5.1.8, “Replication of CURRENT\_USER()”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-current-user).) This is also true for [**VERSION()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_version) and [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand).

For [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now), the binary log includes the timestamp. This means that the value *as returned by the call to this function on the source* is replicated to the replica. To avoid unexpected results when replicating between MySQL servers in different time zones, set the time zone on both source and replica. For more information, see [Section 17.5.1.33, “Replication and Time Zones”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-timezone).

To explain the potential problems when replicating between servers which are in different time zones, suppose that the source is located in New York, the replica is located in Stockholm, and both servers are using local time. Suppose further that, on the source, you create a table **mytable**, perform an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement on this table, and then select from the table, as shown here:

mysql> **CREATE TABLE mytable (mycol TEXT);**

Query OK, 0 rows affected (0.06 sec)

mysql> **INSERT INTO mytable VALUES ( NOW() );**

Query OK, 1 row affected (0.00 sec)

mysql> **SELECT \* FROM mytable;**

+---------------------+

| mycol |

+---------------------+

| 2009-09-01 12:00:00 |

+---------------------+

1 row in set (0.00 sec)

Local time in Stockholm is 6 hours later than in New York; so, if you issue **SELECT NOW()** on the replica at that exact same instant, the value **2009-09-01 18:00:00** is returned. For this reason, if you select from the replica's copy of **mytable** after the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements just shown have been replicated, you might expect **mycol** to contain the value **2009-09-01 18:00:00**. However, this is not the case; when you select from the replica's copy of **mytable**, you obtain exactly the same result as on the source:

mysql> **SELECT \* FROM mytable;**

+---------------------+

| mycol |

+---------------------+

| 2009-09-01 12:00:00 |

+---------------------+

1 row in set (0.00 sec)

Unlike [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now), the [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) function is not replication-safe because it is not affected by **SET TIMESTAMP** statements in the binary log and is nondeterministic if statement-based logging is used. This is not a problem if row-based logging is used.

An alternative is to use the [--sysdate-is-now](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_sysdate-is-now) option to cause [**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) to be an alias for [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now). This must be done on the source and the replica to work correctly. In such cases, a warning is still issued by this function, but can safely be ignored as long as [--sysdate-is-now](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_sysdate-is-now) is used on both the source and the replica.

[**SYSDATE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sysdate) is automatically replicated using row-based replication when using **MIXED** mode, and generates a warning in **STATEMENT** mode.

See also [Section 17.5.1.33, “Replication and Time Zones”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-timezone).

*The following restriction applies to statement-based replication only, not to row-based replication.* The [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock), [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock), [**IS\_FREE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-free-lock), and [**IS\_USED\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_is-used-lock) functions that handle user-level locks are replicated without the replica knowing the concurrency context on the source. Therefore, these functions should not be used to insert into a source table because the content on the replica would differ. For example, do not issue a statement such as **INSERT INTO mytable VALUES(GET\_LOCK(...))**.

These functions are automatically replicated using row-based replication when using **MIXED** mode, and generate a warning in **STATEMENT** mode.

As a workaround for the preceding limitations when statement-based replication is in effect, you can use the strategy of saving the problematic function result in a user variable and referring to the variable in a later statement. For example, the following single-row [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is problematic due to the reference to the [**UUID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_uuid) function:

INSERT INTO t VALUES(UUID());

To work around the problem, do this instead:

SET @my\_uuid = UUID();

INSERT INTO t VALUES(@my\_uuid);

That sequence of statements replicates because the value of **@my\_uuid** is stored in the binary log as a user-variable event prior to the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement and is available for use in the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert).

The same idea applies to multiple-row inserts, but is more cumbersome to use. For a two-row insert, you can do this:

SET @my\_uuid1 = UUID(); @my\_uuid2 = UUID();

INSERT INTO t VALUES(@my\_uuid1),(@my\_uuid2);

However, if the number of rows is large or unknown, the workaround is difficult or impracticable. For example, you cannot convert the following statement to one in which a given individual user variable is associated with each row:

INSERT INTO t2 SELECT UUID(), \* FROM t1;

Within a stored function, [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) replicates correctly as long as it is invoked only once during the execution of the function. (You can consider the function execution timestamp and random number seed as implicit inputs that are identical on the source and replica.)

The [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) and [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) functions are not replicated reliably using statement-based replication. A workaround is to store the result of the function call in a user variable, and then use that in the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement. For example, if you wish to store the result in a table named **mytable**, you might normally do so like this:

SELECT SQL\_CALC\_FOUND\_ROWS FROM mytable LIMIT 1;

INSERT INTO mytable VALUES( FOUND\_ROWS() );

However, if you are replicating **mytable**, you should use [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into), and then store the variable in the table, like this:

SELECT SQL\_CALC\_FOUND\_ROWS INTO @found\_rows FROM mytable LIMIT 1;

INSERT INTO mytable VALUES(@found\_rows);

In this way, the user variable is replicated as part of the context, and applied on the replica correctly.

These functions are automatically replicated using row-based replication when using **MIXED** mode, and generate a warning in **STATEMENT** mode. (Bug #12092, Bug #30244)

**17.5.1.15 Replication and Fractional Seconds Support**

MySQL 8.0 permits fractional seconds for [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) values, with up to microseconds (6 digits) precision. See [Section 11.2.6, “Fractional Seconds in Time Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fractional-seconds).

**17.5.1.16 Replication of Invoked Features**

Replication of invoked features such as user-defined functions (UDFs) and stored programs (stored procedures and functions, triggers, and events) provides the following characteristics:

The effects of the feature are always replicated.

The following statements are replicated using statement-based replication:

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event)

[**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event)

[**DROP EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-event)

[**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure)

[**DROP PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure)

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function)

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function)

[**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger)

[**DROP TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-trigger)

However, the *effects* of features created, modified, or dropped using these statements are replicated using row-based replication.

**Note**

Attempting to replicate invoked features using statement-based replication produces the warning Statement is not safe to log in statement format. For example, trying to replicate a UDF with statement-based replication generates this warning because it currently cannot be determined by the MySQL server whether the UDF is deterministic. If you are absolutely certain that the invoked feature's effects are deterministic, you can safely disregard such warnings.

In the case of [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) and [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event):

The status of the event is set to **SLAVESIDE\_DISABLED** on the replica regardless of the state specified (this does not apply to [**DROP EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-event)).

The source on which the event was created is identified on the replica by its server ID. The **ORIGINATOR** column in [**INFORMATION\_SCHEMA.EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) stores this information. See [Section 26.3.14, “The INFORMATION\_SCHEMA EVENTS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table), and [Section 13.7.7.18, “SHOW EVENTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events), for more information.

The feature implementation resides on the replica in a renewable state so that if the source fails, the replica can be used as the source without loss of event processing.

To determine whether there are any scheduled events on a MySQL server that were created on a different server (that was acting as a source), query the [**INFORMATION\_SCHEMA.EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table in a manner similar to what is shown here:

SELECT EVENT\_SCHEMA, EVENT\_NAME

FROM INFORMATION\_SCHEMA.EVENTS

WHERE STATUS = 'SLAVESIDE\_DISABLED';

Alternatively, you can use the [**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events) statement, like this:

SHOW EVENTS

WHERE STATUS = 'SLAVESIDE\_DISABLED';

When promoting a replica having such events to a source, you must enable each event using [**ALTER EVENT *event\_name* ENABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event), where ***event\_name*** is the name of the event.

If more than one source was involved in creating events on this replica, and you wish to identify events that were created only on a given source having the server ID ***source\_id***, modify the previous query on the [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table to include the **ORIGINATOR** column, as shown here:

SELECT EVENT\_SCHEMA, EVENT\_NAME, ORIGINATOR

FROM INFORMATION\_SCHEMA.EVENTS

WHERE STATUS = 'SLAVESIDE\_DISABLED'

AND ORIGINATOR = '***source\_id***'

You can employ **ORIGINATOR** with the [**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events) statement in a similar fashion:

SHOW EVENTS

WHERE STATUS = 'SLAVESIDE\_DISABLED'

AND ORIGINATOR = '***source\_id***'

Before enabling events that were replicated from the source, you should disable the MySQL Event Scheduler on the replica (using a statement such as **SET GLOBAL event\_scheduler = OFF;**), run any necessary [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statements, restart the server, then re-enable the Event Scheduler on the replica afterward (using a statement such as **SET GLOBAL event\_scheduler = ON;**)-

If you later demote the new source back to being a replica, you must disable manually all events enabled by the [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statements. You can do this by storing in a separate table the event names from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement shown previously, or using [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statements to rename the events with a common prefix such as **replicated\_** to identify them.

If you rename the events, then when demoting this server back to being a replica, you can identify the events by querying the [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table, as shown here:

SELECT CONCAT(EVENT\_SCHEMA, '.', EVENT\_NAME) AS 'Db.Event'

FROM INFORMATION\_SCHEMA.EVENTS

WHERE INSTR(EVENT\_NAME, 'replicated\_') = 1;

**17.5.1.17 Replication of JSON Documents**

Before MySQL 8.0, an update to a JSON column was always written to the binary log as the complete document. In MySQL 8.0, it is possible to log partial updates to JSON documents (see [Partial Updates of JSON Values](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json-partial-updates)), which is more efficient. The logging behavior depends on the format used, as described here:

**Statement-based replication.** JSON partial updates are always logged as partial updates. This cannot be disabled when using statement-based logging.

**Row-based replication.** JSON partial updates are not logged as such by default, but instead are logged as complete documents. To enable logging of partial updates, set [**binlog\_row\_value\_options=PARTIAL\_JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_value_options). If a replication source has this variable set, partial updates received from that source are handled and applied by a replica regardless of the replica's own setting for the variable.

Servers running MySQL 8.0.2 or earlier do not recognize the log events used for JSON partial updates. For this reason, when replicating to such a server from a server running MySQL 8.0.3 or later, **binlog\_row\_value\_options** must be disabled on the source by setting this variable to **''** (empty string). See the description of this variable for more information.

**17.5.1.18 Replication and LIMIT**

Statement-based replication of **LIMIT** clauses in [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements is unsafe since the order of the rows affected is not defined. (Such statements can be replicated correctly with statement-based replication only if they also contain an **ORDER BY** clause.) When such a statement is encountered:

When using **STATEMENT** mode, a warning that the statement is not safe for statement-based replication is now issued.

When using **STATEMENT** mode, warnings are issued for DML statements containing **LIMIT** even when they also have an **ORDER BY** clause (and so are made deterministic). This is a known issue. (Bug #42851)

When using **MIXED** mode, the statement is now automatically replicated using row-based mode.

**17.5.1.19 Replication and LOAD DATA**

**[LOAD DATA](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "load-data" \o "13.2.7 LOAD DATA Statement)** is considered unsafe for statement-based logging (see [Section 17.2.1.3, “Determination of Safe and Unsafe Statements in Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-safe-unsafe)). When [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, the statement is logged in row-based format. When [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, note that [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) does not generate a warning, unlike other unsafe statements.

If you use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) with [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format), each replica on which the changes are to be applied creates a temporary file containing the data. The replica then uses a [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement to apply the changes. This temporary file is not encrypted, even if binary log encryption is active on the source, If encryption is required, use row-based or mixed binary logging format instead, for which replicas do not create the temporary file.

If a **PRIVILEGE\_CHECKS\_USER** account has been used to help secure the replication channel (see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks)), it is strongly recommended that you log [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) operations using row-based binary logging ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)). If **REQUIRE\_ROW\_FORMAT** is set for the channel, row-based binary logging is required. With this logging format, the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege is not needed to execute the event, so do not give the **PRIVILEGE\_CHECKS\_USER** account this privilege. If you need to recover from a replication error involving a **LOAD DATA INFILE** operation logged in statement format, and the replicated event is trusted, you could grant the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to the **PRIVILEGE\_CHECKS\_USER** account temporarily, removing it after the replicated event has been applied.

When [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) reads log events for [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements logged in statement-based format, a generated local file is created in a temporary directory. These temporary files are not automatically removed by [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) or any other MySQL program. If you do use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements with statement-based binary logging, you should delete the temporary files yourself after you no longer need the statement log. For more information, see [Section 4.6.9, “mysqlbinlog — Utility for Processing Binary Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).

**17.5.1.20 Replication and max\_allowed\_packet**

**[max\_allowed\_packet](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\server-administration.html" \l "sysvar_max_allowed_packet)** sets an upper limit on the size of any single message between the MySQL server and clients, including replicas. If you are replicating large column values (such as might be found in [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns) and [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) is too small on the source, the source fails with an error, and the replica shuts down the replication I/O thread. If [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) is too small on the replica, this also causes the replica to stop the I/O thread.

Row-based replication currently sends all columns and column values for updated rows from the source to the replica, including values of columns that were not actually changed by the update. This means that, when you are replicating large column values using row-based replication, you must take care to set [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) large enough to accommodate the largest row in any table to be replicated, even if you are replicating updates only, or you are inserting only relatively small values.

On a multi-threaded replica (with [**slave\_parallel\_workers > 0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers)), ensure that the [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) system variable is set to a value equal to or greater than the setting for the [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) system variable on the source. The default setting for [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max), 128M, is twice the default setting for [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet), which is 64M. [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) limits the packet size that the source can send, but the addition of an event header can produce a binary log event exceeding this size. Also, in row-based replication, a single event can be significantly larger than the [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) size, because the value of [**max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_allowed_packet) only limits each column of the table.

The replica actually accepts packets up to the limit set by its [**slave\_max\_allowed\_packet**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_max_allowed_packet) setting, which defaults to the maximum setting of 1GB, to prevent a replication failure due to a large packet. However, the value of [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) controls the memory that is made available on the replica to hold incoming packets. The specified memory is shared among all the replica worker queues.

The value of [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) is a soft limit, and if an unusually large event (consisting of one or multiple packets) exceeds this size, the transaction is held until all the replica workers have empty queues, and then processed. All subsequent transactions are held until the large transaction has been completed. So although unusual events larger than [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) can be processed, the delay to clear the queues of all the replica workers and the wait to queue subsequent transactions can cause lag on the replica and decreased concurrency of the replica workers. [**slave\_pending\_jobs\_size\_max**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_pending_jobs_size_max) should therefore be set high enough to accommodate most expected event sizes.

**17.5.1.21 Replication and MEMORY Tables**

When a replication source server shuts down and restarts, its [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables become empty. To replicate this effect to replicas, the first time that the source uses a given [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) table after startup, it logs an event that notifies replicas that the table must be emptied by writing a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or (from MySQL 8.0.22) [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement for that table to the binary log. This generated event is identifiable by a comment in the binary log, and if GTIDs are in use on the server, it has a GTID assigned. The statement is always logged in statement format, even if the binary logging format is set to **ROW**, and it is written even if **read\_only** or **super\_read\_only** mode is set on the server. Note that the replica still has outdated data in a [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) table during the interval between the source's restart and its first use of the table. To avoid this interval when a direct query to the replica could return stale data, you can set the [**init\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_file) system variable to name a file containing statements that populate the **MEMORY** table on the source at startup.

When a replica server shuts down and restarts, its [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables become empty. This causes the replica to be out of synchrony with the source and may lead to other failures or cause the replica to stop:

Row-format updates and deletes received from the source may fail with **Can't find record in '*memory\_table*'**.

Statements such as [**INSERT INTO ... SELECT FROM *memory\_table***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) may insert a different set of rows on the source and replica.

The replica also writes a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or (from MySQL 8.0.22) [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement to its own binary log, which is passed on to any downstream replicas, causing them to empty their own [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables.

The safe way to restart a replica that is replicating [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables is to first drop or delete all rows from the [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables on the source and wait until those changes have replicated to the replica. Then it is safe to restart the replica.

An alternative restart method may apply in some cases. When [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format), you can prevent the replica from stopping if you set [**slave\_exec\_mode=IDEMPOTENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) before you start the replica again. This allows the replica to continue to replicate, but its [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables still differ from those on the source. This is acceptable if the application logic is such that the contents of [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables can be safely lost (for example, if the [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables are used for caching). [**slave\_exec\_mode=IDEMPOTENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) applies globally to all tables, so it may hide other replication errors in non-[**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables.

(The method just described is not applicable in NDB Cluster, where [**slave\_exec\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_exec_mode) is always **IDEMPOTENT**, and cannot be changed.)

The size of [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables is limited by the value of the [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) system variable, which is not replicated (see [Section 17.5.1.39, “Replication and Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-variables)). A change in **max\_heap\_table\_size** takes effect for **MEMORY** tables that are created or updated using [**ALTER TABLE ... ENGINE = MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) following the change, or for all [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables following a server restart. If you increase the value of this variable on the source without doing so on the replica, it becomes possible for a table on the source to grow larger than its counterpart on the replica, leading to inserts that succeed on the source but fail on the replica with Table is full errors. This is a known issue (Bug #48666). In such cases, you must set the global value of [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) on the replica as well as on the source, then restart replication. It is also recommended that you restart both the source and replica MySQL servers, to ensure that the new value takes complete (global) effect on each of them.

See [Section 16.3, “The MEMORY Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine), for more information about [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables.

**17.5.1.22 Replication of the mysql System Schema**

Data modification statements made to tables in the **mysql** schema are replicated according to the value of [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format); if this value is **MIXED**, these statements are replicated using row-based format. However, statements that would normally update this information indirectly—such [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke), and statements manipulating triggers, stored routines, and views—are replicated to replicas using statement-based replication.

**17.5.1.23 Replication and the Query Optimizer**

It is possible for the data on the source and replica to become different if a statement is written in such a way that the data modification is nondeterministic; that is, left up the query optimizer. (In general, this is not a good practice, even outside of replication.) Examples of nondeterministic statements include [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements that use **LIMIT** with no **ORDER BY** clause; see [Section 17.5.1.18, “Replication and LIMIT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-limit), for a detailed discussion of these.

**17.5.1.24 Replication and Partitioning**

Replication is supported between partitioned tables as long as they use the same partitioning scheme and otherwise have the same structure, except where an exception is specifically allowed (see [Section 17.5.1.9, “Replication with Differing Table Definitions on Source and Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-differing-tables)).

Replication between tables that have different partitioning is generally not supported. This because statements (such as [**ALTER TABLE ... DROP PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations)) that act directly on partitions in such cases might produce different results on the source and the replica. In the case where a table is partitioned on the source but not on the replica, any statements that operate on partitions on the source's copy of the replica fail on the replica. When the replica's copy of the table is partitioned but the source's copy is not, statements that act directly on partitions cannot be run on the source without causing errors there. To avoid stopping replication or creating inconsistencies between the source and replica, always ensure that a table on the source and the corresponding replicated table on the replica are partitioned in the same way.

**17.5.1.25 Replication and REPAIR TABLE**

When used on a corrupted or otherwise damaged table, it is possible for the [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statement to delete rows that cannot be recovered. However, any such modifications of table data performed by this statement are not replicated, which can cause source and replica to lose synchronization. For this reason, in the event that a table on the source becomes damaged and you use [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) to repair it, you should first stop replication (if it is still running) before using [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), then afterward compare the source's and replica's copies of the table and be prepared to correct any discrepancies manually, before restarting replication.

**17.5.1.26 Replication and Reserved Words**

You can encounter problems when you attempt to replicate from an older source to a newer replica and you make use of identifiers on the source that are reserved words in the newer MySQL version running on the replica. For example, a table column named **rank** on a MySQL 5.7 source that is replicating to a MySQL 8.0 replica could cause a problem because **RANK** is a reserved word beginning in MySQL 8.0.

Replication can fail in such cases with Error 1064 You have an error in your SQL syntax..., *even if a database or table named using the reserved word or a table having a column named using the reserved word is excluded from replication*. This is due to the fact that each SQL event must be parsed by the replica prior to execution, so that the replica knows which database object or objects would be affected. Only after the event is parsed can the replica apply any filtering rules defined by [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db), [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table), [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db), and [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table).

To work around the problem of database, table, or column names on the source which would be regarded as reserved words by the replica, do one of the following:

Use one or more [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements on the source to change the names of any database objects where these names would be considered reserved words on the replica, and change any SQL statements that use the old names to use the new names instead.

In any SQL statements using these database object names, write the names as quoted identifiers using backtick characters (**`**).

For listings of reserved words by MySQL version, see [Keywords and Reserved Words in MySQL 8.0](https://dev.mysql.com/doc/mysqld-version-reference/en/keywords-8-0.html), in the *MySQL Server Version Reference*. For identifier quoting rules, see [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

**17.5.1.27 Replication and Row Searches**

When a replica using row-based replication format applies an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation, it must search the relevant table for the matching rows. The algorithm used to carry out this process uses one of the table's indexes to carry out the search as the first choice, and a hash table if there are no suitable indexes.

The algorithm first assesses the available indexes in the table definition to see if there is any suitable index to use, and if there are multiple possibilities, which index is the best fit for the operation. The algorithm ignores the following types of index:

Fulltext indexes.

Hidden indexes.

Generated indexes.

Multi-valued indexes.

Any index where the before-image of the row event does not contain all the columns of the index.

If there are no suitable indexes after ruling out these index types, the algorithm does not use an index for the search. If there are suitable indexes, one index is selected from the candidates, in the following priority order:

A primary key.

A unique index where every column in the index has a NOT NULL attribute. If more than one such index is available, the algorithm chooses the leftmost of these indexes.

Any other index. If more than one such index is available, the algorithm chooses the leftmost of these indexes.

If the algorithm is able to select a primary key or a unique index where every column in the index has a **NOT NULL** attribute, it uses this index to iterate over the rows in the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation. For each row in the row event, the algorithm looks up the row in the index to locate the table record to update. If no matching record is found, it returns the error ER\_KEY\_NOT\_FOUND and stops the replication applier thread.

If the algorithm was not able to find a suitable index, or was only able to find an index that was non-unique or contained nulls, a hash table is used to assist in identifying the table records. The algorithm creates a hash table containing the rows in the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation, with the key as the full before-image of the row. The algorithm then iterates over all the records in the target table, using the selected index if it found one, or else performing a full table scan. For each record in the target table, it determines whether that row exists in the hash table. If the row is found in the hash table, the record in the target table is updated, and the row is deleted from the hash table. When all the records in the target table have been checked, the algorithm verifies whether the hash table is now empty. If there are any unmatched rows remaining in the hash table, the algorithm returns the error ER\_KEY\_NOT\_FOUND and stops the replication applier thread.

The [**slave\_rows\_search\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_rows_search_algorithms) system variable was previously used to control how rows are searched for matches. The use of this system variable is now deprecated, because the default setting, which uses an index scan followed by a hash scan as described above, is optimal for performance and works correctly in all scenarios.

**17.5.1.28 Replication and Source or Replica Shutdowns**

It is safe to shut down a replication source server and restart it later. When a replica loses its connection to the source, the replica tries to reconnect immediately and retries periodically if that fails. The default is to retry every 60 seconds. This may be changed with the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). A replica also is able to deal with network connectivity outages. However, the replica notices the network outage only after receiving no data from the source for [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) seconds. If your outages are short, you may want to decrease [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout). See [Section 17.4.2, “Handling an Unexpected Halt of a Replica”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-unexpected-replica-halt).

An unclean shutdown (for example, a crash) on the source side can result in the source's binary log having a final position less than the most recent position read by the replica, due to the source's binary log file not being flushed. This can cause the replica not to be able to replicate when the source comes back up. Setting [**sync\_binlog=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_binlog) in the source server's my.cnf file helps to minimize this problem because it causes the source to flush its binary log more frequently. For the greatest possible durability and consistency in a replication setup using **InnoDB** with transactions, you should also set [**innodb\_flush\_log\_at\_trx\_commit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit). With this setting, the contents of the **InnoDB** redo log buffer are written out to the log file at each transaction commit and the log file is flushed to disk. Note that the durability of transactions is still not guaranteed with this setting, because operating systems or disk hardware may tell [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) that the flush-to-disk operation has taken place, even though it has not.

Shutting down a replica cleanly is safe because it keeps track of where it left off. However, be careful that the replica does not have temporary tables open; see [Section 17.5.1.31, “Replication and Temporary Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-temptables). Unclean shutdowns might produce problems, especially if the disk cache was not flushed to disk before the problem occurred:

For transactions, the replica commits and then updates relay-log.info. If an unexpected exit occurs between these two operations, relay log processing proceeds further than the information file indicates and the replica re-executes the events from the last transaction in the relay log after it has been restarted.

A similar problem can occur if the replica updates relay-log.info but the server host crashes before the write has been flushed to disk. To minimize the chance of this occurring, set [**sync\_relay\_log\_info=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log_info) in the replica my.cnf file. Setting [**sync\_relay\_log\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sync_relay_log_info) to 0 causes no writes to be forced to disk and the server relies on the operating system to flush the file from time to time.

The fault tolerance of your system for these types of problems is greatly increased if you have a good uninterruptible power supply.

**17.5.1.29 Replica Errors During Replication**

If a statement produces the same error (identical error code) on both the source and the replica, the error is logged, but replication continues.

If a statement produces different errors on the source and the replica, the replication SQL thread terminates, and the replica writes a message to its error log and waits for the database administrator to decide what to do about the error. This includes the case that a statement produces an error on the source or the replica, but not both. To address the issue, connect to the replica manually and determine the cause of the problem. [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) is useful for this. Then fix the problem and run [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica). For example, you might need to create a nonexistent table before you can start the replica again.

**Note**

If a temporary error is recorded in the replica's error log, you do not necessarily have to take any action suggested in the quoted error message. Temporary errors should be handled by the client retrying the transaction. For example, if the replication SQL thread records a temporary error relating to a deadlock, you do not need to restart the transaction manually on the replica, unless the replication SQL thread subsequently terminates with a nontemporary error message.

If this error code validation behavior is not desirable, some or all errors can be masked out (ignored) with the [--slave-skip-errors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_slave-skip-errors) option.

For nontransactional storage engines such as **MyISAM**, it is possible to have a statement that only partially updates a table and returns an error code. This can happen, for example, on a multiple-row insert that has one row violating a key constraint, or if a long update statement is killed after updating some of the rows. If that happens on the source, the replica expects execution of the statement to result in the same error code. If it does not, the replication SQL thread stops as described previously.

If you are replicating between tables that use different storage engines on the source and replica, keep in mind that the same statement might produce a different error when run against one version of the table, but not the other, or might cause an error for one version of the table, but not the other. For example, since **MyISAM** ignores foreign key constraints, an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement accessing an **InnoDB** table on the source might cause a foreign key violation but the same statement performed on a **MyISAM** version of the same table on the replica would produce no such error, causing replication to stop.

**17.5.1.30 Replication and Server SQL Mode**

Using different server SQL mode settings on the source and the replica may cause the same [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements to be handled differently on the source and the replica, leading the source and replica to diverge. For best results, you should always use the same server SQL mode on the source and on the replica. This advice applies whether you are using statement-based or row-based replication.

If you are replicating partitioned tables, using different SQL modes on the source and the replica is likely to cause issues. At a minimum, this is likely to cause the distribution of data among partitions to be different in the source's and replica's copies of a given table. It may also cause inserts into partitioned tables that succeed on the source to fail on the replica.

For more information, see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

**17.5.1.31 Replication and Temporary Tables**

In MySQL 8.0, when [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **ROW** or **MIXED**, statements that exclusively use temporary tables are not logged on the source, and therefore the temporary tables are not replicated. Statements that involve a mix of temporary and nontemporary tables are logged on the source only for the operations on nontemporary tables, and the operations on temporary tables are not logged. This means that there are never any temporary tables on the replica to be lost in the event of an unplanned shutdown by the replica. For more information about row-based replication and temporary tables, see [Row-based logging of temporary tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-usage-temptables).

When [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set to **STATEMENT**, operations on temporary tables are logged on the source and replicated on the replica, provided that the statements involving temporary tables can be logged safely using statement-based format. In this situation, loss of replicated temporary tables on the replica can be an issue. In statement-based replication mode, [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**DROP TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements cannot be used inside a transaction, procedure, function, or trigger when GTIDs are in use on the server (that is, when the [**enforce\_gtid\_consistency**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_enforce_gtid_consistency) system variable is set to **ON**). They can be used outside these contexts when GTIDs are in use, provided that [**autocommit=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is set.

Because of the differences in behavior between row-based or mixed replication mode and statement-based replication mode regarding temporary tables, you cannot switch the replication format at runtime, if the change applies to a context (global or session) that contains any open temporary tables. For more details, see the description of the [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) option.

**Safe replica shutdown when using temporary tables.** In statement-based replication mode, temporary tables are replicated except in the case where you stop the replica server (not just the replication threads) and you have replicated temporary tables that are open for use in updates that have not yet been executed on the replica. If you stop the replica server, the temporary tables needed by those updates are no longer available when the replica is restarted. To avoid this problem, do not shut down the replica while it has temporary tables open. Instead, use the following procedure:

Issue a **STOP REPLICA | SLAVE SQL\_THREAD** statement.

Use [**SHOW STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status) to check the value of the [**Slave\_open\_temp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_open_temp_tables) variable.

If the value is not 0, restart the replication SQL thread with **START REPLICA | SLAVE SQL\_THREAD** and repeat the procedure later.

When the value is 0, issue a [**mysqladmin shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) command to stop the replica.

**Temporary tables and replication options.** By default, with statement-based replication, all temporary tables are replicated; this happens whether or not there are any matching [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db), [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table), or [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table) options in effect. However, the [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table) and [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) options are honored for temporary tables. The exception is that to enable correct removal of temporary tables at the end of a session, a replica always replicates a **DROP TEMPORARY TABLE IF EXISTS** statement, regardless of any exclusion rules that would normally apply for the specified table.

A recommended practice when using statement-based replication is to designate a prefix for exclusive use in naming temporary tables that you do not want replicated, then employ a [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) option to match that prefix. For example, you might give all such tables names beginning with **norep** (such as **norepmytable**, **norepyourtable**, and so on), then use [--replicate-wild-ignore-table=norep%](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) to prevent them from being replicated.

**17.5.1.32 Replication Retries and Timeouts**

The global system variable [**slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries) sets the maximum number of times for applier threads on a single-threaded or multithreaded replica to automatically retry failed transactions before stopping. Transactions are automatically retried when the SQL thread fails to execute them because of an **InnoDB** deadlock, or when the transaction's execution time exceeds the **InnoDB** [**innodb\_lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_lock_wait_timeout) value. If a transaction has a non-temporary error that prevents it from succeeding, it is not retried.

The default setting for [**slave\_transaction\_retries**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_transaction_retries) is 10, meaning that a failing transaction with an apparently temporary error is retried 10 times before the applier thread stops. Setting the variable to 0 disables automatic retrying of transactions. On a multithreaded replica, the specified number of transaction retries can take place on all applier threads of all channels. The Performance Schema table [**replication\_applier\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-table) shows the total number of transaction retries that took place on each replication channel, in the **COUNT\_TRANSACTIONS\_RETRIES** column.

The process of retrying transactions can cause lag on a replica or on a Group Replication group member, which can be configured as a single-threaded or multithreaded replica. The Performance Schema table [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) shows detailed information on transaction retries by the applier threads on a single-threaded or multithreaded replica. This data includes timestamps showing how long it took the applier thread to apply the last transaction from start to finish (and when the transaction currently in progress was started), and how long this was after the commit on the original source and the immediate source. The data also shows the number of retries for the last transaction and the transaction currently in progress, and enables you to identify the transient errors that caused the transactions to be retried. You can use this information to see whether transaction retries are the cause of replication lag, and investigate the root cause of the failures that led to the retries.

**17.5.1.33 Replication and Time Zones**

By default, source and replica servers assume that they are in the same time zone. If you are replicating between servers in different time zones, the time zone must be set on both source and replica. Otherwise, statements depending on the local time on the source are not replicated properly, such as statements that use the [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) or [**FROM\_UNIXTIME()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_from-unixtime) functions.

Verify that your combination of settings for the system time zone ([**system\_time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_system_time_zone)), server current time zone (the global value of [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone)), and per-session time zones (the session value of [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone)) on the source and replica is producing the correct results. In particular, if the [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone) system variable is set to the value **SYSTEM**, indicating that the server time zone is the same as the system time zone, this can cause the source and replica to apply different time zones. For example, a source could write the following statement in the binary log:

SET @@session.time\_zone='SYSTEM';

If this source and its replica have a different setting for their system time zones, this statement can produce unexpected results on the replica, even if the replica's global [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone) value has been set to match the source's. For an explanation of MySQL Server's time zone settings, and how to change them, see [Section 5.1.15, “MySQL Server Time Zone Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#time-zone-support).

See also [Section 17.5.1.14, “Replication and System Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-functions).

**17.5.1.34 Replication and Transaction Inconsistencies**

Inconsistencies in the sequence of transactions that have been executed from the relay log can occur depending on your replication configuration. This section explains how to avoid inconsistencies and solve any problems they cause.

The following types of inconsistencies can exist:

*Half-applied transactions*. A transaction which updates non-transactional tables has applied some but not all of its changes.

*Gaps*. A gap in the externalized transaction set appears when, given an ordered sequence of transactions, a transaction that is later in the sequence is applied before some other transaction that is prior in the sequence. Gaps can only appear when using a multithreaded replica. To avoid gaps occurring, set [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order). Up to and including MySQL 8.0.18, this setting requires that binary logging ([**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin)) and replica update logging ([**log\_slave\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates)) are also enabled, which are the default settings from MySQL 8.0. From MySQL 8.0.19, binary logging and replica update logging are not required on the replica to set [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order), and can be disabled if wanted. In all releases, setting [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) requires that [**slave\_parallel\_type**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_type) is set to **LOGICAL\_CLOCK**, which is *not* the default setting. Note that in some specific situations, as listed in the description for [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order), setting [**slave\_preserve\_commit\_order=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) cannot preserve commit order on the replica, so in these cases gaps might still appear in the sequence of transactions that have been executed from the replica's relay log.

*Source binary log position lag*. Even in the absence of gaps, it is possible that transactions after **Exec\_master\_log\_pos** have been applied. That is, all transactions up to point **N** have been applied, and no transactions after **N** have been applied, but **Exec\_master\_log\_pos** has a value smaller than **N**. In this situation, **Exec\_master\_log\_pos** is a “low-water mark” of the transactions applied, and lags behind the position of the most recently applied transaction. This can only happen on multithreaded replicas. Enabling [**slave\_preserve\_commit\_order**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_preserve_commit_order) does not prevent source binary log position lag.

The following scenarios are relevant to the existence of half-applied transactions, gaps, and source binary log position lag:

While replication threads are running, there may be gaps and half-applied transactions.

[**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) shuts down. Both clean and unclean shutdown abort ongoing transactions and may leave gaps and half-applied transactions.

[**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) of replication threads (the SQL thread when using a single-threaded replica, the coordinator thread when using a multithreaded replica). This aborts ongoing transactions and may leave gaps and half-applied transactions.

Error in applier threads. This may leave gaps. If the error is in a mixed transaction, that transaction is half-applied. When using a multithreaded replica, workers which have not received an error complete their queues, so it may take time to stop all threads.

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) when using a multithreaded replica. After issuing [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica), the replica waits for any gaps to be filled and then updates **Exec\_master\_log\_pos**. This ensures it never leaves gaps or source binary log position lag, unless any of the cases above applies, in other words, before [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) completes, either an error happens, or another thread issues [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill), or the server restarts. In these cases, [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) returns successfully.

If the last transaction in the relay log is only half-received and the multithreaded replica's coordinator thread has started to schedule the transaction to a worker, then [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) waits up to 60 seconds for the transaction to be received. After this timeout, the coordinator gives up and aborts the transaction. If the transaction is mixed, it may be left half-completed.

[**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) when using a single-threaded replica. If the ongoing transaction only updates transactional tables, it is rolled back and [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) stops immediately. If the ongoing transaction is mixed, [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) waits up to 60 seconds for the transaction to complete. After this timeout, it aborts the transaction, so it may be left half-completed.

The global variable [**rpl\_stop\_slave\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_stop_slave_timeout) is unrelated to the process of stopping the replication threads. It only makes the client that issues [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) return to the client, but the replication threads continue to try to stop.

If a replication channel has gaps, it has the following consequences:

The replica database is in a state that may never have existed on the source.

The field **Exec\_master\_log\_pos** in [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) is only a “low-water mark”. In other words, transactions appearing before the position are guaranteed to have committed, but transactions after the position may have committed or not.

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) and [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statements for that channel fail with an error, unless the applier threads are running and the statement only sets receiver options.

If [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started with [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery), no recovery is done for that channel, and a warning is printed.

If [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) is used with [--dump-slave](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_dump-slave), it does not record the existence of gaps; thus it prints [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) with **RELAY\_LOG\_POS** set to the “low-water mark” position in **Exec\_master\_log\_pos**.

After applying the dump on another server, and starting the replication threads, transactions appearing after the position are replicated again. Note that this is harmless if GTIDs are enabled (however, in that case it is not recommended to use [--dump-slave](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysqldump_dump-slave)).

If a replication channel has source binary log position lag but no gaps, cases 2 to 5 above apply, but case 1 does not.

The source binary log position information is persisted in binary format in the internal table **mysql.slave\_worker\_info**. [**START REPLICA | SLAVE [SQL\_THREAD]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) always consults this information so that it applies only the correct transactions. This remains true even if [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) has been changed to 0 before [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), and even if [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) is used with **UNTIL** clauses. [**START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) only applies as many transactions as needed in order to fill in the gaps. If [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) is used with **UNTIL** clauses that tell it to stop before it has consumed all the gaps, then it leaves remaining gaps.

**Warning**

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) removes the relay logs and resets the replication position. Thus issuing [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) on a replica with gaps means the replica loses any information about the gaps, without correcting the gaps.

When GTID-based replication is in use, from MySQL 5.7.28 a multithreaded replica checks first whether **MASTER\_AUTO\_POSITION** is set to **ON**, and if it is, omits the step of calculating the transactions that should be skipped or not skipped. In that situation, the old relay logs are not required for the recovery process.

**17.5.1.35 Replication and Transactions**

**Mixing transactional and nontransactional statements within the same transaction.** In general, you should avoid transactions that update both transactional and nontransactional tables in a replication environment. You should also avoid using any statement that accesses both transactional (or temporary) and nontransactional tables and writes to any of them.

The server uses these rules for binary logging:

If the initial statements in a transaction are nontransactional, they are written to the binary log immediately. The remaining statements in the transaction are cached and not written to the binary log until the transaction is committed. (If the transaction is rolled back, the cached statements are written to the binary log only if they make nontransactional changes that cannot be rolled back. Otherwise, they are discarded.)

For statement-based logging, logging of nontransactional statements is affected by the [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) system variable. When this variable is **OFF** (the default), logging is as just described. When this variable is **ON**, logging occurs immediately for nontransactional statements occurring anywhere in the transaction (not just initial nontransactional statements). Other statements are kept in the transaction cache and logged when the transaction commits. [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) has no effect for row-format or mixed-format binary logging.

**Transactional, nontransactional, and mixed statements.** To apply those rules, the server considers a statement nontransactional if it changes only nontransactional tables, and transactional if it changes only transactional tables. A statement that references both nontransactional and transactional tables and updates *any* of the tables involved is considered a “mixed” statement. Mixed statements, like transactional statements, are cached and logged when the transaction commits.

A mixed statement that updates a transactional table is considered unsafe if the statement also performs either of the following actions:

Updates or reads a temporary table

Reads a nontransactional table and the transaction isolation level is less than REPEATABLE\_READ

A mixed statement following the update of a transactional table within a transaction is considered unsafe if it performs either of the following actions:

Updates any table and reads from any temporary table

Updates a nontransactional table and [**binlog\_direct\_non\_transactional\_updates**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_direct_non_transactional_updates) is OFF

For more information, see [Section 17.2.1.3, “Determination of Safe and Unsafe Statements in Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-safe-unsafe).

**Note**

A mixed statement is unrelated to mixed binary logging format.

In situations where transactions mix updates to transactional and nontransactional tables, the order of statements in the binary log is correct, and all needed statements are written to the binary log even in case of a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). However, when a second connection updates the nontransactional table before the first connection transaction is complete, statements can be logged out of order because the second connection update is written immediately after it is performed, regardless of the state of the transaction being performed by the first connection.

**Using different storage engines on source and replica.** It is possible to replicate transactional tables on the source using nontransactional tables on the replica. For example, you can replicate an **InnoDB** source table as a **MyISAM** replica table. However, if you do this, there are problems if the replica is stopped in the middle of a [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) ... [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) block because the replica restarts at the beginning of the [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) block.

It is also safe to replicate transactions from [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables on the source to transactional tables, such as tables that use the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine, on the replica. In such cases, an [**AUTOCOMMIT=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) statement issued on the source is replicated, thus enforcing **AUTOCOMMIT** mode on the replica.

When the storage engine type of the replica is nontransactional, transactions on the source that mix updates of transactional and nontransactional tables should be avoided because they can cause inconsistency of the data between the source transactional table and the replica nontransactional table. That is, such transactions can lead to source storage engine-specific behavior with the possible effect of replication going out of synchrony. MySQL does not issue a warning about this, so extra care should be taken when replicating transactional tables from the source to nontransactional tables on the replicas.

**Changing the binary logging format within transactions.** The [**binlog\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) and [**binlog\_checksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_checksum) system variables are read-only as long as a transaction is in progress.

Every transaction (including [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) transactions) is recorded in the binary log as though it starts with a [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement, and ends with either a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) or a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement. This is even true for statements affecting tables that use a nontransactional storage engine (such as [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine)).

**Note**

For restrictions that apply specifically to XA transactions, see [Section 13.3.8.3, “Restrictions on XA Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-restrictions).

**17.5.1.36 Replication and Triggers**

With statement-based replication, triggers executed on the source also execute on the replica. With row-based replication, triggers executed on the source do not execute on the replica. Instead, the row changes on the source resulting from trigger execution are replicated and applied on the replica.

This behavior is by design. If under row-based replication the replica applied the triggers as well as the row changes caused by them, the changes would in effect be applied twice on the replica, leading to different data on the source and the replica.

If you want triggers to execute on both the source and the replica, perhaps because you have different triggers on the source and replica, you must use statement-based replication. However, to enable replica-side triggers, it is not necessary to use statement-based replication exclusively. It is sufficient to switch to statement-based replication only for those statements where you want this effect, and to use row-based replication the rest of the time.

A statement invoking a trigger (or function) that causes an update to an **AUTO\_INCREMENT** column is not replicated correctly using statement-based replication. MySQL 8.0 marks such statements as unsafe. (Bug #45677)

A trigger can have triggers for different combinations of trigger event ([**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete)) and action time (**BEFORE**, **AFTER**), and multiple triggers are permitted.

For brevity, “multiple triggers” here is shorthand for “multiple triggers that have the same trigger event and action time.”

***Upgrades.*** Multiple triggers are not supported in versions earlier than MySQL 5.7. If you upgrade servers in a replication topology that use a version earlier than MySQL 5.7, upgrade the replicas first and then upgrade the source. If an upgraded replication source server still has old replicas using MySQL versions that do not support multiple triggers, an error occurs on those replicas if a trigger is created on the source for a table that already has a trigger with the same trigger event and action time.

***Downgrades.*** If you downgrade a server that supports multiple triggers to an older version that does not, the downgrade has these effects:

For each table that has triggers, all trigger definitions are in the .TRG file for the table. However, if there are multiple triggers with the same trigger event and action time, the server executes only one of them when the trigger event occurs. For information about **.TRG** files, see the Table Trigger Storage section of the MySQL Server Doxygen documentation, available at <https://dev.mysql.com/doc/index-other.html>.

If triggers for the table are added or dropped subsequent to the downgrade, the server rewrites the table's .TRG file. The rewritten file retains only one trigger per combination of trigger event and action time; the others are lost.

To avoid these problems, modify your triggers before downgrading. For each table that has multiple triggers per combination of trigger event and action time, convert each such set of triggers to a single trigger as follows:

For each trigger, create a stored routine that contains all the code in the trigger. Values accessed using **NEW** and **OLD** can be passed to the routine using parameters. If the trigger needs a single result value from the code, you can put the code in a stored function and have the function return the value. If the trigger needs multiple result values from the code, you can put the code in a stored procedure and return the values using **OUT** parameters.

Drop all triggers for the table.

Create one new trigger for the table that invokes the stored routines just created. The effect for this trigger is thus the same as the multiple triggers it replaces.

**17.5.1.37 Replication and TRUNCATE TABLE**

**[TRUNCATE TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "truncate-table" \o "13.1.37 TRUNCATE TABLE Statement)** is normally regarded as a DML statement, and so would be expected to be logged and replicated using row-based format when the binary logging mode is **ROW** or **MIXED**. However this caused issues when logging or replicating, in **STATEMENT** or **MIXED** mode, tables that used transactional storage engines such as [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) when the transaction isolation level was **READ COMMITTED** or **READ UNCOMMITTED**, which precludes statement-based logging.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) is treated for purposes of logging and replication as DDL rather than DML so that it can be logged and replicated as a statement. However, the effects of the statement as applicable to [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and other transactional tables on replicas still follow the rules described in [Section 13.1.37, “TRUNCATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) governing such tables. (Bug #36763)

**17.5.1.38 Replication and User Name Length**

The maximum length for user names in MySQL 8.0 is 32 characters. Replication of user names longer than 16 characters fails when the replica runs a version of MySQL previous to 5.7, because those versions support only shorter user names. This occurs only when replicating from a newer source to an older replica, which is not a recommended configuration.

**17.5.1.39 Replication and Variables**

System variables are not replicated correctly when using **STATEMENT** mode, except for the following variables when they are used with session scope:

[**auto\_increment\_increment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_increment)

[**auto\_increment\_offset**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_auto_increment_offset)

[**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client)

[**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection)

[**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database)

[**character\_set\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_server)

[**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection)

[**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database)

[**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server)

[**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks)

[**identity**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_identity)

[**last\_insert\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_last_insert_id)

[**lc\_time\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lc_time_names)

[**pseudo\_thread\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_pseudo_thread_id)

[**sql\_auto\_is\_null**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null)

[**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone)

[**timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_timestamp)

[**unique\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_unique_checks)

When **MIXED** mode is used, the variables in the preceding list, when used with session scope, cause a switch from statement-based to row-based logging. See [Section 5.4.4.3, “Mixed Binary Logging Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-mixed).

[**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) is also replicated except for the [**NO\_DIR\_IN\_CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_dir_in_create) mode; the replica always preserves its own value for [**NO\_DIR\_IN\_CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_dir_in_create), regardless of changes to it on the source. This is true for all replication formats.

However, when [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) parses a **SET @@sql\_mode = *mode*** statement, the full ***mode*** value, including [**NO\_DIR\_IN\_CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_dir_in_create), is passed to the receiving server. For this reason, replication of such a statement may not be safe when **STATEMENT** mode is in use.

The [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) system variable is not replicated, regardless of the logging mode; this is intended to facilitate replication between different storage engines.

The [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is not replicated. In addition, the enabling this variable has different effects with regard to temporary tables, table locking, and the [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) statement in different MySQL versions.

The [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) system variable is not replicated. Increasing the value of this variable on the source without doing so on the replica can lead eventually to Table is full errors on the replica when trying to execute [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements on a [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) table on the source that is thus permitted to grow larger than its counterpart on the replica. For more information, see [Section 17.5.1.21, “Replication and MEMORY Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-memory).

In statement-based replication, session variables are not replicated properly when used in statements that update tables. For example, the following sequence of statements does not insert the same data on the source and the replica:

SET max\_join\_size=1000;

INSERT INTO mytable VALUES(@@max\_join\_size);

This does not apply to the common sequence:

SET time\_zone=...;

INSERT INTO mytable VALUES(CONVERT\_TZ(..., ..., @@time\_zone));

Replication of session variables is not a problem when row-based replication is being used, in which case, session variables are always replicated safely. See [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).

The following session variables are written to the binary log and honored by the replica when parsing the binary log, regardless of the logging format:

[**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode)

[**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks)

[**unique\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_unique_checks)

[**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client)

[**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection)

[**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database)

[**collation\_server**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_server)

[**sql\_auto\_is\_null**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null)

**Important**

Even though session variables relating to character sets and collations are written to the binary log, replication between different character sets is not supported.

To help reduce possible confusion, we recommend that you always use the same setting for the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable on both source and replica, especially when you are running MySQL on platforms with case-sensitive file systems. The [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting can only be configured when initializing the server.

**17.5.1.40 Replication and Views**

Views are always replicated to replicas. Views are filtered by their own name, not by the tables they refer to. This means that a view can be replicated to the replica even if the view contains a table that would normally be filtered out by replication-ignore-table rules. Care should therefore be taken to ensure that views do not replicate table data that would normally be filtered for security reasons.

Replication from a table to a same-named view is supported using statement-based logging, but not when using row-based logging. Trying to do so when row-based logging is in effect causes an error.

**17.5.2 Replication Compatibility Between MySQL Versions**

MySQL supports replication from one release series to the next higher release series. For example, you can replicate from a source running MySQL 5.6 to a replica running MySQL 5.7, from a source running MySQL 5.7 to a replica running MySQL 8.0, and so on. However, you might encounter difficulties when replicating from an older source to a newer replica if the source uses statements or relies on behavior no longer supported in the version of MySQL used on the replica. For example, foreign key names longer than 64 characters are no longer supported from MySQL 8.0.

The use of more than two MySQL Server versions is not supported in replication setups involving multiple sources, regardless of the number of source or replica MySQL servers. This restriction applies not only to release series, but to version numbers within the same release series as well. For example, if you are using a chained or circular replication setup, you cannot use MySQL 8.0.1, MySQL 8.0.2, and MySQL 8.0.4 concurrently, although you could use any two of these releases together.

**Important**

It is strongly recommended to use the most recent release available within a given MySQL release series because replication (and other) capabilities are continually being improved. It is also recommended to upgrade sources and replicas that use early releases of a release series of MySQL to GA (production) releases when the latter become available for that release series.

From MySQL 8.0.14, the server version is recorded in the binary log for each transaction for the server that originally committed the transaction ([**original\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_original_server_version)), and for the server that is the immediate source of the current server in the replication topology ([**immediate\_server\_version**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_immediate_server_version)).

Replication from newer sources to older replicas might be possible, but is generally not supported. This is due to a number of factors:

**Binary log format changes.** The binary log format can change between major releases. While we attempt to maintain backward compatibility, this is not always possible. A source might also have optional features enabled that are not understood by older replicas, such as binary log transaction compression, where the resulting compressed transaction payloads cannot be read by a replica at a release before MySQL 8.0.20.

This also has significant implications for upgrading replication servers; see [Section 17.5.3, “Upgrading a Replication Setup”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-upgrade), for more information.

For more information about row-based replication, see [Section 17.2.1, “Replication Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-formats).

**SQL incompatibilities.** You cannot replicate from a newer source to an older replica using statement-based replication if the statements to be replicated use SQL features available on the source but not on the replica.

However, if both the source and the replica support row-based replication, and there are no data definition statements to be replicated that depend on SQL features found on the source but not on the replica, you can use row-based replication to replicate the effects of data modification statements even if the DDL run on the source is not supported on the replica.

For more information on potential replication issues, see [Section 17.5.1, “Replication Features and Issues”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features).

**17.5.3 Upgrading a Replication Setup**

When you upgrade servers that participate in a replication setup, the procedure for upgrading depends on the current server versions and the version to which you are upgrading. This section provides information about how upgrading affects replication. For general information about upgrading MySQL, see [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading)

When you upgrade a source to 8.0 from an earlier MySQL release series, you should first ensure that all the replicas of this source are using the same 8.0.x release. If this is not the case, you should first upgrade the replicas. To upgrade each replica, shut it down, upgrade it to the appropriate 8.0.x version, restart it, and restart replication. Relay logs created by the replica after the upgrade are in 8.0 format.

Changes affecting operations in strict SQL mode ([**STRICT\_TRANS\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_strict_trans_tables) or [**STRICT\_ALL\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_strict_all_tables)) may result in replication failure on an upgraded replica. If you use statement-based logging ([**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)), if a replica is upgraded before the source, the source executes statements which succeed there but which may fail on the replica and so cause replication to stop. To deal with this, stop all new statements on the source and wait until the replicas catch up, then upgrade the replicas. Alternatively, if you cannot stop new statements, temporarily change to row-based logging on the source ([**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format)) and wait until all replicas have processed all binary logs produced up to the point of this change, then upgrade the replicas.

The default character set has changed from **latin1** to **utf8mb4** in MySQL 8.0. In a replicated setting, when upgrading from MySQL 5.7 to 8.0, it is advisable to change the default character set back to the character set used in MySQL 5.7 before upgrading. After the upgrade is completed, the default character set can be changed to **utf8mb4**. Assuming that the previous defaults were used, one way to preserve them is to start the server with these lines in the my.cnf file:

[mysqld]

character\_set\_server=latin1

collation\_server=latin1\_swedish\_ci

After the replicas have been upgraded, shut down the source, upgrade it to the same 8.0.x release as the replicas, and restart it. If you had temporarily changed the source to row-based logging, change it back to statement-based logging. The 8.0 source is able to read the old binary logs written prior to the upgrade and to send them to the 8.0 replicas. The replicas recognize the old format and handle it properly. Binary logs created by the source subsequent to the upgrade are in 8.0 format. These too are recognized by the 8.0 replicas.

In other words, when upgrading to MySQL 8.0, the replicas must be MySQL 8.0 before you can upgrade the source to 8.0. Note that downgrading from 8.0 to older versions does not work so simply: You must ensure that any 8.0 binary log or relay log has been fully processed, so that you can remove it before proceeding with the downgrade.

Some upgrades may require that you drop and re-create database objects when you move from one MySQL series to the next. For example, collation changes might require that table indexes be rebuilt. Such operations, if necessary, are detailed at [Section 2.11.4, “Changes in MySQL 8.0”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading-from-previous-series). It is safest to perform these operations separately on the replicas and the source, and to disable replication of these operations from the source to the replica. To achieve this, use the following procedure:

Stop all the replicas and upgrade them. Restart them with the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable, so that they do not connect to the source. Perform any table repair or rebuilding operations needed to re-create database objects, such as use of **REPAIR TABLE** or **ALTER TABLE**, or dumping and reloading tables or triggers.

Disable the binary log on the source. To do this without restarting the source, execute a **SET sql\_log\_bin = OFF** statement. Alternatively, stop the source and restart it with the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option. If you restart the source, you might also want to disallow client connections. For example, if all clients connect using TCP/IP, enable the [**skip\_networking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_skip_networking) system variable when you restart the source.

With the binary log disabled, perform any table repair or rebuilding operations needed to re-create database objects. The binary log must be disabled during this step to prevent these operations from being logged and sent to the replicas later.

Re-enable the binary log on the source. If you set [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) to **OFF** earlier, execute a **SET sql\_log\_bin = ON** statement. If you restarted the source to disable the binary log, restart it without [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin), and without enabling the [**skip\_networking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_skip_networking) system variable so that clients and replicas can connect.

Restart the replicas, this time without the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option or [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable.

If you are upgrading an existing replication setup from a version of MySQL that does not support global transaction identifiers to a version that does, you should not enable GTIDs on either the source or the replica before making sure that the setup meets all the requirements for GTID-based replication. See [Section 17.1.3.4, “Setting Up Replication Using GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-howto), which contains information about converting existing replication setups to use GTID-based replication.

Prior to MySQL 8.0.16, when the server is running with global transaction identifiers (GTIDs) enabled ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), do not enable binary logging by [**mysql\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-upgrade) (the [--write-binlog](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_upgrade_write-binlog) option). As of MySQL 8.0.16, the server performs the entire MySQL upgrade procedure, but disables binary logging during the upgrade, so there is no issue.

It is not recommended to load a dump file when GTIDs are enabled on the server ([**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode)), if your dump file includes system tables. [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) issues DML instructions for the system tables which use the non-transactional MyISAM storage engine, and this combination is not permitted when GTIDs are enabled. Also be aware that loading a dump file from a server with GTIDs enabled, into another server with GTIDs enabled, causes different transaction identifiers to be generated.

**17.5.4 Troubleshooting Replication**

If you have followed the instructions but your replication setup is not working, the first thing to do is *check the error log for messages*. Many users have lost time by not doing this soon enough after encountering problems.

If you cannot tell from the error log what the problem was, try the following techniques:

Verify that the source has binary logging enabled by issuing a [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) statement. Binary logging is enabled by default. If binary logging is enabled, **Position** is nonzero. If binary logging is not enabled, verify that you are not running the source with any settings that disable binary logging, such as the [--skip-log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option.

Verify that the [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) system variable was set at startup on both the source and replica and that the ID value is unique on each server.

Verify that the replica is running. Use [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) to check whether the **Replica\_IO\_Running** and **Replica\_SQL\_Running** values are both **Yes**. If not, verify the options that were used when starting the replica server. For example, the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) command line option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable, prevents the replication threads from starting until you issue a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement.

If the replica is running, check whether it established a connection to the source. Use [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist), find the I/O and SQL threads and check their **State** column to see what they display. See [Section 17.2.3, “Replication Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation-details). If the I/O thread state says **Connecting to master**, check the following:

Verify the privileges for the replication user on the source.

Check that the host name of the source is correct and that you are using the correct port to connect to the source. The port used for replication is the same as used for client network communication (the default is **3306**). For the host name, ensure that the name resolves to the correct IP address.

Check the configuration file to see whether the [**skip\_networking**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_skip_networking) system variable has been enabled on the source or replica to disable networking. If so, comment the setting or remove it.

If the source has a firewall or IP filtering configuration, ensure that the network port being used for MySQL is not being filtered.

Check that you can reach the source by using **ping** or **traceroute**/**tracert** to reach the host.

If the replica was running previously but has stopped, the reason usually is that some statement that succeeded on the source failed on the replica. This should never happen if you have taken a proper snapshot of the source, and never modified the data on the replica outside of the replication threads. If the replica stops unexpectedly, it is a bug or you have encountered one of the known replication limitations described in [Section 17.5.1, “Replication Features and Issues”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features). If it is a bug, see [Section 17.5.5, “How to Report Replication Bugs or Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-bugs), for instructions on how to report it.

If a statement that succeeded on the source refuses to run on the replica, try the following procedure if it is not feasible to do a full database resynchronization by deleting the replica's databases and copying a new snapshot from the source:

Determine whether the affected table on the replica is different from the source table. Try to understand how this happened. Then make the replica's table identical to the source's and run [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica).

If the preceding step does not work or does not apply, try to understand whether it would be safe to make the update manually (if needed) and then ignore the next statement from the source.

If you decide that the replica can skip the next statement from the source, issue the following statements:

mysql> **SET GLOBAL sql\_slave\_skip\_counter = *N*;**

mysql> **START SLAVE;**

Or from MySQL 8.0.22:

mysql> **START REPLICA;**

The value of ***N*** should be 1 if the next statement from the source does not use **AUTO\_INCREMENT** or [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id). Otherwise, the value should be 2. The reason for using a value of 2 for statements that use **AUTO\_INCREMENT** or [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) is that they take two events in the binary log of the source.

See also [SET GLOBAL sql\_slave\_skip\_counter Statement](https://dev.mysql.com/doc/refman/5.6/en/set-global-sql-slave-skip-counter.html).

If you are sure that the replica started out perfectly synchronized with the source, and that no one has updated the tables involved outside of the replication threads, then presumably the discrepancy is the result of a bug. If you are running the most recent version of MySQL, please report the problem. If you are running an older version, try upgrading to the latest production release to determine whether the problem persists.

**17.5.5 How to Report Replication Bugs or Problems**

When you have determined that there is no user error involved, and replication still either does not work at all or is unstable, it is time to send us a bug report. We need to obtain as much information as possible from you to be able to track down the bug. Please spend some time and effort in preparing a good bug report.

If you have a repeatable test case that demonstrates the bug, please enter it into our bugs database using the instructions given in [Section 1.6, “How to Report Bugs or Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#bug-reports). If you have a “phantom” problem (one that you cannot duplicate at will), use the following procedure:

Verify that no user error is involved. For example, if you update the replica outside of the replication threads, the data goes out of synchrony, and you can have unique key violations on updates. In this case, the replication thread stops and waits for you to clean up the tables manually to bring them into synchrony. *This is not a replication problem. It is a problem of outside interference causing replication to fail.*

Ensure that the replica is running with binary logging enabled (the [**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) system variable), and with the [--log-slave-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_slave_updates) option enabled, which causes the replica to log the updates that it receives from the source into its own binary logs. These settings are the defaults.

Save all evidence before resetting the replication state. If we have no information or only sketchy information, it becomes difficult or impossible for us to track down the problem. The evidence you should collect is:

All binary log files from the source

All binary log files from the replica

The output of [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) from the source at the time you discovered the problem

The output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) from the replica at the time you discovered the problem

Error logs from the source and the replica

Use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) to examine the binary logs. The following should be helpful to find the problem statement. ***log\_file*** and ***log\_pos*** are the **Master\_Log\_File** and **Read\_Master\_Log\_Pos** values from [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status).

shell> **mysqlbinlog --start-position=*log\_pos* *log\_file* | head**

After you have collected the evidence for the problem, try to isolate it as a separate test case first. Then enter the problem with as much information as possible into our bugs database using the instructions at [Section 1.6, “How to Report Bugs or Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#bug-reports).

|  |  |  |
| --- | --- | --- |
| [Prev](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html) | Up | [Next](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html) |
| Chapter 16 Alternative Storage Engines | [Home](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\index.html) | Chapter 18 Group Replication |