概论

MySQL™软件提供了一个非常快速、多线程、多用户和强大的SQL（结构化查询语言）数据库服务器。MySQL服务器的目的是用于关键任务、重负荷的生产系统以及嵌入大规模部署的软件。MySQL是甲骨文公司和/或其附属机构的商标。或其附属公司的商标，未经甲骨文公司明确的书面授权，客户不得使用。其他名称可能是其各自所有者的商标.

MySQL软件是双授权的。用户可以选择将MySQL软件作为开放源码产品在GNU通用公共许可证的条款下使用。用户可以选择根据GNU通用公共许可证（http://www.fsf.org/licenses/）的条款将MySQL软件作为开源产品使用，或者从甲骨文公司购买标准的商业许可证。可以从Oracle购买一个标准的商业许可。参见<http://www.mysql.com/company/legal/> 许可/了解更多关于我们许可政策的信息。

以下列表描述了本手册中一些特别值得关注的章节。

* 关于MySQL数据库服务器功能的讨论，见第1.2.2节 "MySQL的主要特性"。
* 关于MySQL新特性的概述，见第1.3节 "MySQL 8.0的新内容"。关于每个版本的关于每个版本的变化信息，请参见发行说明。
* 关于安装说明，见第2章，安装和升级MySQL。关于升级MySQL的信息，见第2.11节 "升级MySQL"。
* 关于MySQL数据库服务器的教程介绍，见第3章，教程。
* 有关配置和管理MySQL服务器的信息，见第5章，MySQL服务器管理。
* 关于MySQL的安全信息，见第6章，安全。
* 有关设置复制服务器的信息，见第17章，复制。
* 关于MySQL企业版的信息，这是一个具有高级功能和管理工具的商业MySQL版本，见第30章。
* 有关MySQL企业版的信息，请参见第30章，MySQL企业版
* 关于经常被问到的有关MySQL数据库服务器及其功能的一些问题的答案，见附录A，MySQL 8.0常见问题。及其功能的问题，请参见附录A，MySQL 8.0常见问题。
* 有关新功能和错误修复的历史，请参见发行说明。

注意事项：

要报告问题或错误，请使用第1.6节的说明。"如何报告错误或问题"。如果你发现MySQL服务器中的安全漏洞，请立即让我们知道，请发送电子邮件到<secalert\_us@oracle.com>。例外。支持客户应报告所有问题，包括安全漏洞，都应报告给Oracle支持。

## 1.1关于本手册

这是MySQL数据库系统8.0-8.0.26版的参考手册。

MySQL 8.0各次要版本之间的差异在本文中以发布号（8.0.x）为参照进行说明。版本号（8.0.x）。关于许可证信息，请参见法律声明。

本手册不打算用于MySQL软件的旧版本，因为MySQL 8.0与MySQL 8.0之间存在许多功能和其他方面的差异。

由于MySQL 8.0与以前的版本在功能和其他方面有许多不同之处，本手册不打算用于旧版本的MySQL软件。如果你正在使用较早的

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。

由于本手册作为参考，它并不提供关于SQL或关系数据库概念的一般指导。数据库的概念。它也不教你如何使用你的操作系统或命令行解释器。

MySQL数据库软件正在不断开发中，参考手册也经常更新。也经常更新。该手册的最新版本可以在网上以可搜索的形式获得，网址是https://dev.mysql.com/doc/。其他格式也可在那里找到，包括可下载的HTML和PDF版本。

MySQL的源代码本身包含使用Doxygen编写的内部文档。生成的生成的Doxygen内容可在https://dev.mysql.com/doc/index-other.html。它也可以也可以使用第2.9.10节 "生成MySQL Doxygen文档内容 "中的说明，从MySQL源分发中本地生成这些内容。第2.9.10节，"生成MySQL Doxygen文档内容"。

如果你有关于使用MySQL的问题，请加入MySQL社区Slack，或在我们的论坛上提问；见MySQL论坛上的MySQL社区支持。如果你有关于添加或如果你有关于手册本身的补充或修正的建议，请将它们发送到<http://www.mysql.com/company/contact/>。

**排版和语法惯例**

本手册使用了某些排版惯例

* 这种风格的文本用于SQL语句；数据库、表和列名；程序清单和源代码；以及环境变量。列表和源代码；以及环境变量。例如。"要重新加载授予表，请使用FLUSH PRIVILEGES语句"。
* 这种风格的文本表示你在例子中输入的内容**。**
* 这种风格的文本表示可执行程序和脚本的名称，例子有mysql（MySQL命令行客户端程序）和mysqld（MySQL服务器可执行程序）**。**
* 这种风格的文本用于变量输入，你应该用一个你自己选择的值来代替它。自己选择的值**。**
* 这种风格的文本用于强调**。** 这种风格的文本用于表的标题和传达特别强烈的强调**。这种风格的文本用于表示一个影响程序执行方式的程序选项执行，或提供程序以某种方式运行所需的信息。例如。"--host选项（简称-h）告诉mysql客户端程序它应该连接的MySQL服务器的主机名或IP地址它应该连接到的MySQL服务器的主机名或IP地址"。**
* **文件名和目录名是这样写的。"全局的my.cnf文件位于/etc目录中。"**
* **字符序列是这样写的。"要指定一个通配符，请使用'%'字符"**

当显示的命令是要在一个特定的程**序中执行时，**在命令前面显示的提示符表明要使用哪个命令。命令前显示的提示符表明要使用哪条命令。例如，shell>表示你从登录的shell中执行的命令，root-shell>类似，但应以root身份执行。mysql>表示从mysql客户端程序中执行的语句

shell> **type a shell command here**

root-shell> **type a shell command as *root* here**

mysql> **type a mysql statement here**

在一些地区，不同的系统可能会被区分开来，以表明命令应该在两个不同的环境中执行。在两个不同的环境中执行。例如，在复制工作中，命令可能以源和复制为前缀。

source> **type a mysql command on the replication source here**

replica> **type a mysql command on the replica here**

shell "是你的命令解释器。在Unix中，这通常是一个程序，如sh、csh或bash。在Windows中，相应的程序是command.com或cmd.exe，通常在一个控制台窗口中运行。

当你输入一个例子中的命令或语句时，不要输入例子中的提示符。例子中显示的提示。

数据库、表和列的名称经常必须被替换成语句。为了表示这种，本手册使用db\_name、tbl\_name和col\_name来表示这种替换是必要的。例如，你可能会看到这样的语句。

mysql> **SELECT *col\_name* FROM *db\_name*.*tbl\_name*;**

这意味着，如果你要输入一个类似的语句，你将提供你自己的数据库、表。和列名，也许像这样。

mysql> **SELECT author\_name FROM biblio\_db.author\_list;**

SQL关键字不区分大小写，可以用任何字母大小写来书写。本手册使用大写字母。

在语法描述中，方括号（"["和"]"）表示可选的词或子句。例如，在下面的语句中，IF EXISTS是可选的。

DROP TABLE [IF EXISTS] ***tbl\_name***

当一个语法元素由多个备选方案组成时，这些备选方案由竖条（"|"）分隔。条（"|"）分隔。当可以从一组选项中选择一个成员时，这些选项被列在方括号（"["和"]"）。

TRIM([[BOTH | LEADING | TRAILING] [***remstr***] FROM] ***str***)

当必须从一组选择中选择一个成员时，备选方案被列在大括号内（"{"和"}"）。)

{DESCRIBE | DESC} ***tbl\_name*** [***col\_name*** | ***wild***]

省略号(....)表示省略了语句中的某一节，通常是为了提供更简短的，通常是为更复杂的语法提供一个简短的版本。例如，SELECT ... INTO OUTFILE是SELECT语句的速记形式，它在其他部分之后有一个INTO OUTFILE子句。语句的其他部分之后有一个INTO OUTFILE子句的SELECT语句的简称。

省略号也可以表示语句中前面的句法元素可以重复。在下面的例子中，可以给出多个reset\_option值，在第一个值之后的每一个值都用逗号表示。前面有逗号。

RESET ***reset\_option*** [,***reset\_option***

设置shell变量的命令使用Bourne shell的语法来显示。例如，在Bourne shell中，设置CC环境变量和运行configure命令的顺序是这样的来设置CC环境变量和运行configure命令，在Bourne shell中是这样的语法。

shell> **CC=gcc ./configure**

如果你使用的是 csh 或 tcsh，你必须以某种不同的方式发布命令。

shell> **setenv CC gcc**

shell> **./configure**

**手册撰写说明**

参考手册的源文件是以DocBook XML格式编写的。HTML版本和其他格式自动生成，主要使用DocBook XSL样式表。关于DocBook的信息有关DocBook的信息，请参见http://docbook.org/

本手册最初由David Axmark和Michael "Monty" Widenius编写。它由MySQL文档团队维护。它由MySQL文档团队维护，该团队由Chris Cole、Paul DuBois、Margaret Fisher、Edward，Gilmore, Stefan Hinz, David Moss, Philip Olson, Daniel Price, Daniel So, and Jon Stephens.

## 1.2 MySQL数据库管理系统概览

### 1.2.1 MySQL是什么？

MySQL是最受欢迎的开源SQL数据库管理系统，由甲骨文公司开发、发布和支持。

MySQL网站（http://www.mysql.com/）提供关于MySQL软件的最新信息。

* MySQL是数据库管理系统

数据库是一个结构化的数据集合。数据集可以是任何东西，从简单的购物清单到一个图片库或一个公司网络中的大量信息。要添加、访问和处理存储在计算机数据库中的数据，你需要一个数据库管理系统，例如MySQL服务器这样的数据库管理系统。由于计算机非常善于处理大量的数据，所以数据库管理系统在计算机领域发挥着核心作用。数据库管理系统在计算中发挥着核心作用，作为独立的实用程序，或作为其他应用程序的一部分。

* MySQL数据库是关系型的

关系型数据库将数据存储在不同的表中，而不是将所有的数据放在一个大的存储空间。数据库结构被组织成物理文件，并为速度而优化。逻辑模型逻辑模型，包括数据库、表、视图、行和列等对象，提供了一个灵活的编程环境。你可以设置不同数据之间的关系的规则字段之间的关系，如一对一、一对多、唯一、必需或可选，以及不同表之间的 "指针"。不同表之间的 "指针"。数据库强制执行这些规则，因此，在一个设计良好的数据库中，你的应用程序永远不会看到不一致的、重复的和不完整的数据。

MySQL "的SQL部分代表 "结构化查询语言"。SQL是最常见的用于访问数据库的标准化语言。取决于你的编程环境。你可以直接输入SQL（例如，生成报告），将SQL语句嵌入到用另一种语言编写的代码中，或使用一种特定语言的代码。用另一种语言编写的代码中，或者使用隐藏SQL语法的特定语言API。

SQL是由ANSI/ISO SQL标准定义的。自1986年以来，SQL标准一直在不断发展并有多个版本。在本手册中，"SQL-92 "是指1992年发布的标准。"SQL:1999 "是指1999年发布的标准，而"SQL:2003 "是指当前版本的标准。是指该标准的当前版本。我们使用 "SQL标准 "这个短语来指当前版本的SQL标准在任何时候

* MySQL是开源的

开源意味着任何人都有可能使用和修改该软件。任何人都可以从互联网上下载MySQL软件并使用它，而无需支付任何费用。如果你愿意，你可以研究源代码并根据自己的需要进行修改。MySQL软件使用GPL(GNU General Public License)http://www.fsf.org/licenses/，来定义你在不同情况下可以和不可以做什么在不同情况下可以和不可以做什么。如果你对GPL感到不舒服，或者需要将MySQL代码嵌入到一个商业应用中，你可以将MySQL代码嵌入到一个商业应用中。MySQL代码嵌入商业应用，你可以向我们购买商业许可版本。更多信息见MySQL许可概述(<http://www.mysql.com/company/legal/licensing/> )

* MySQL具有高效、可靠、弹性和易用的特性

如果这是你正在寻找的，你应该试一试。MySQL服务器可以舒适地运行在台式机或笔记本电脑上，与你的其他应用程序、网络服务器等一起运行，只需要很少或不需要注意。如果你把一整台机器用于MySQL，你可以调整设置，以利用所有的内存、CPU功率、以及其他的功能。所有可用的内存、CPU功率和I/O容量。MySQL还可以扩展到联网的机器集群。联网的机器集群。

MySQL服务器最初是为了处理大型数据库而开发的，比现有的解决方案快得多。几年来，它已被成功地用于高要求的生产环境中。几年来一直成功地用于高要求的生产环境。尽管在不断的发展中，MySQL服务器今天提供了丰富和有用的功能集。功能。它的连接性、速度和安全性使MySQL服务器非常适用于访问互联网上的数据库。

* MySQL可以是客户端/服务器架构也可以是嵌入式架构

MySQL数据库软件是一个客户/服务器系统，由一个多线程的SQL服务器组成。支持不同的后端，几个不同的客户端程序和库，管理工具。以及广泛的应用编程接口（API）。

我们还提供MySQL服务器作为一个嵌入式的多线程库，你可以将其链接到你的以获得更小、更快、更易于管理的独立产品。

* 有大量贡献的MySQL软件可用。

MySQL服务器有一套与我们的用户密切合作开发的实用功能。它是很可能你最喜欢的应用程序或语言支持MySQL数据库服务器。

MySQL的官方发音是 "My Ess Que Ell"（不是 "my sequel"），但我们并不介意你发音为 "my sequel "或其他一些本地化的方式。

### 1.2.2 MySQL的主要特性

本节介绍了MySQL数据库软件的一些重要特征。在大多数方面，该路线图适用于MySQL的所有版本。有关在特定系列基础上引入MySQL的特性的信息关于在特定系列基础上引入MySQL的特性，请参见相应的《手册》中的 "简明扼要 "部分。

* MySQL 8.0: Section 1.3, “What Is New in MySQL 8.0”
* MySQL 5.7: What Is New in MySQL 5.7
* MySQL 5.6: What Is New in MySQL 5.6

**内部和可移植性**

* 用C和C++编写。
* 用广泛的不同的编译器进行测试。
* 在许多不同的平台上工作。见https://www.mysql.com/support/supportedplatforms/数据库.html。
* 为了便于移植，使用CMake进行配置。
* 用Purify（一个商业的内存泄漏检测器）以及Valgrind（一个GPL工具）进行测试。(<http://developer.kde.org/~sewardj/>)。
* 使用具有独立模块的多层服务器设计。
* 设计成使用内核线程的完全多线程，如果有多个CPU的话，可以很容易地使用它们。可用。
* 提供事务和非事务存储引擎。
* 使用非常快的B树磁盘表（MyISAM），带有索引压缩。旨在使添加其他存储引擎变得相对容易。如果你想为内部数据库提供一个内部数据库的SQL接口。
* 使用一个非常快的基于线程的内存分配系统。
* 使用优化的嵌套循环连接，执行非常快的连接。
* 实施内存哈希表，作为临时表使用。
* 使用高度优化的类库执行SQL函数，其速度应尽可能快。通常，在查询初始化之后，根本没有内存分配。
* 将服务器作为一个单独的程序提供给客户/服务器网络环境使用，并作为一个可以嵌入（链接）的库。一个可以嵌入（链接）到独立应用程序的库。这样的应用程序可以被用于孤立地使用，或在没有网络的环境中使用。

**数据类型**

* 许多数据类型：1、2、3、4和8字节长的有符号/无符号整数、FLOAT、DOUBLE、CHAR。VARCHAR, BINARY, VARBINARY, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET,ENUM，以及OpenGIS空间类型。参见第11章，数据类型。
* 固定长度和可变长度的字符串类型。

**SQL语句和函数**

* 在查询的SELECT列表和WHERE子句中支持全部操作符和函数。举例来说

mysql> **SELECT CONCAT(first\_name, ' ', last\_name)**

-> **FROM citizen**

-> **WHERE income/dependents > 10000 AND age > 30;**

* 完全支持SQL GROUP BY和ORDER BY条款。支持分组函数（COUNT(),AVG(), STD(), SUM(), MAX(), MIN(), 和 GROUP\_CONCAT()
* 支持标准SQL和ODBC的LEFT OUTER JOIN和RIGHT OUTER JOIN。语法。
* 支持标准SQL所要求的对表和列的别名。
* 支持DELETE, INSERT, REPLACE, 和UPDATE，以返回被改变（影响）的行数。改变（影响）的行数，或者通过在连接服务器时设置一个标志来返回匹配的行数。连接到服务器时，通过设置一个标志来返回匹配的行数。
* 支持MySQL特定的SHOW语句，以检索关于数据库、存储引擎、表和索引的信息。检索数据库、存储引擎、表和索引的信息。支持INFORMATION\_SCHEMA数据库，根据标准SQL实现
* EXPLAIN语句，显示优化器如何解决一个查询。
* 函数名与表或列名的独立性。例如，ABS是一个有效的列名。唯一的限制是，对于一个函数调用，在函数名和后面的"("之间不允许有空格。名称和后面的"("之间不能有空格。参见第9.3节，"关键词和保留字"。
* 你可以在同一语句中引用不同数据库的表

**安全**

* **一个非常灵活和安全的权限和密码系统，并能实现基于主机的验证。**
* **当你连接到服务器时，通过对所有密码流量进行加密来保证密码安全。**

**可扩展性和局限性**

* **对大型数据库的支持。我们使用MySQL服务器的数据库包含5000万条记录。我们也知道有些用户使用MySQL服务器，其中有20万个表和大约5,000,000,000行。**
* **每张表最多支持64个索引。每个索引可以由1到16列或部分列组成。InnoDB表的最大索引宽度是767字节或3072字节。参见第15.22节，"InnoDB的限制"。MyISAM表的最大索引宽度是1000字节。参见第16.2节 "MyISAM存储引擎"。索引可以使用CHAR、VARCHAR、BLOB或者TEXT列类型的列的前缀。**

**连接性**

* **客户端可以使用几种协议连接到MySQL服务器**
  + **客户端可以在任何平台上使用TCP/IP套接字进行连接。**
  + **在Windows系统上，如果服务器在启动时启用了named\_pipe系统变量，客户可以使用命名的管道进行连接。如果启动时启用shared\_memory系统变量，Windows服务器也支持共享内存连接。客户端可以通过使用--protocol=memory选项，通过共享内存进行连接。**
  + **在Unix系统上，客户端可以使用Unix域套接字文件进行连接。**
* **MySQL客户端程序可以用许多语言编写。用C语言编写的客户端库可用于用C或C++编写的客户端，或用于任何提供C绑定的语言。**
* **C、C++、Eiffel、Java、Perl、PHP、Python、Ruby和Tcl的API可用，使MySQL客户端可以用许多语言编写。参见第29章，连接器和API。**
* **连接器/ODBC（MyODBC）接口为使用ODBC（开放数据库连接）连接的客户端程序提供MySQL支持。例如，你可以使用MS Access来连接到你的MySQL服务器。客户端可以在Windows或Unix上运行。连接器/ODBC源是可用的。所有的ODBC 2.5功能都支持，还有许多其他功能。参见《MySQL Connector/ODBC开发指南》。**
* **Connector/J接口为使用JDBC连接的Java客户端程序提供MySQL支持。客户端可以在Windows或Unix上运行。Connector/J的源代码是可用的。参见《MySQL Connector/J 5.1开发者指南》。**
* **MySQL Connector/NET使开发人员能够轻松地创建需要与MySQL进行安全、高性能数据连接的.NET应用程序。它实现了所需的ADO.NET接口并集成到ADO.NET感知工具中。开发人员可以使用他们选择的.NET语言构建应用程序。MySQL Connector/NET是一个完全管理的ADO.NET驱动程序，由100%纯C#编写。请参阅MySQL Connector/NET开发者指南。**

**本地化**

* **服务器可以向客户提供多种语言的错误信息。见第10.12节，"设置错误信息语言"。**
* **完全支持几种不同的字符集，包括latin1 (cp1252), german, big5, ujis, 几个Unicode字符集，以及其他。例如，斯堪的纳维亚字符 "å"、"ä "和 "ö "在表和列名中是允许的。**
* **所有数据都保存在所选择的字符集中。**
* **排序和比较是根据默认的字符集和排序方式进行的。当MySQL服务器启动时，可以改变这一点（见第10.3.2节，"服务器字符集和排序方式"）。要看一个非常高级的排序的例子，请看捷克的排序代码。MySQL服务器支持许多不同的字符集，可以在编译时和运行时指定。**
* **服务器时区可以动态改变，个别客户可以指定自己的时区。见第5.1.15节，"MySQL服务器时区支持"。**

**客户端和工具**

* **MySQL包括几个客户端和实用程序。这些程序包括命令行程序，如mysqldump和mysqladmin，以及图形程序，如MySQL Workbench。**
* **MySQL服务器内置了对SQL语句的支持，以检查、优化和修复表。这些语句可以从命令行通过mysqlcheck客户端获得。MySQL还包括myisamchk，这是一个非常快速的命令行工具，用于对MyISAM表执行这些操作。见第4章，MySQL程序。**
* **可以用--help或--?选项调用MySQL程序，以获得在线帮助。**

### **1.2.3 MySQL的历史**

我们一开始打算使用mSQL数据库系统，用我们自己的快速低级（ISAM）例程连接到我们的表。然而，经过一些测试，我们得出结论，mSQL的速度和灵活性都不足以满足我们的需求。这导致了我们的数据库有了一个新的SQL接口，但其API接口几乎与mSQL相同。这个API的设计是为了使为使用mSQL而编写的第三方代码能够容易地移植到MySQL上使用。

MySQL是以联合创始人Monty Widenius的女儿My命名的**。**

MySQL海豚（我们的标志）的名字是 "Sakila"，这是从我们的 "为海豚起名 "比赛中用户建议的大量名字中选出的。获奖的名字是由Ambrose Twebaze提交的，他是来自非洲埃斯瓦蒂尼（原斯威士兰）的开源软件开发者。据Ambrose说，女性的名字Sakila起源于埃斯瓦蒂尼的当地语言SiSwati。萨基拉也是坦桑尼亚阿鲁沙的一个小镇的名字，靠近安布罗斯的原籍国乌干达**。**

## **1.3 MySQL 8.0中的新内容**

本节总结了在MySQL 8.0中增加、废弃和删除的内容。一部分列出了在MySQL 8.0中增加、废弃或删除的MySQL服务器选项和变量；见第1.4节 "MySQL 8.0中增加、废弃或删除的服务器和状态变量及选项**"**

**在MySQL 8.0中增加的功能**

* 数据字典**。 现在，M**ySQL纳入了一个支持事务**的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，字典数据被存储在元数据文件和非事务性的表中。更多信息，见第14章，MySQL数据字典。**
* 原子数据定义语句（Atomic DDL）**.原子DDL语句将与DDL操作相关的数据字典更新、存储引擎操作和二进制日志写入合并为一个单一的原子事务。欲了解更多信息，请参见第13.1.1节，"原子数据定义语句支持"。**
* 升级程序**。 以前，在安装新版本的MySQL后，MySQL服务器会在下次启动时自动升级数据字典表，之后DBA要手动调用mysql\_upgrade来升级mysql模式中的系统表，以及其他模式中的对象，如sys模式和用户模式。**

**从MySQL 8.0.16开始，服务器会执行以前由mysql\_upgrade处理的任务。在安装了新的MySQL版本后，服务器现在会在下次启动时自动执行所有必要的升级任务，而不依赖于DBA调用mysql\_upgrade。此外，服务器会更新帮助表的内容（这是mysql\_upgrade没有做到的）。一个新的-upgrade服务器选项提供了对服务器如何执行自动数据字典和服务器升级操作的控制。更多信息，见第2.11.3节 "MySQL升级过程中升级的内容"。**

* 安全和账户管理**。 添加这些增强功能是为了提高安全性，并使DBA在账户管理方面具有更大的灵活性。**
  + **mysql系统数据库中的grant表现在是InnoDB（事务性）表。以前，这些是MyISAM（非事务型）表。grant表存储引擎的变化是对账户管理语句行为的相应变化。以前，一个名为多个用户的账户管理语句（如CREATE USER或DROP USER）可能对一些用户成功，对另一些用户失败。现在，每个语句都是事务性的，要么对所有指定的用户成功，要么在发生任何错误时回滚，没有任何影响。如果语句成功了，就会被写入二进制日志，但如果失败了就不会；在这种情况下，就会发生回滚，不做任何改变。更多信息，请参阅第13.1.1节，"原子数据定义语句支持"。**
  + **一个新的caching\_sha2\_password认证插件已经可用。与sha256\_password插件一样，caching\_sha2\_password实现了SHA-256密码散列，但使用缓存来解决连接时的延迟问题。它还支持更多的传输协议，并且不需要与OpenSSL链接以获得基于RSA密钥对的密码交换功能。参见第6.4.1.2节，"缓存SHA-2可插拔认证"。**

**caching\_sha2\_password和sha256\_password认证插件比mysql\_native\_password插件提供更安全的密码加密，而且caching\_sha2\_password比sha256\_password提供更好的性能。由于caching\_sha2\_password的这些优越的安全和性能特性，它现在是首选的认证插件，也是默认的认证插件，而不是mysql\_native\_password。关于这种默认插件的变化对服务器操作以及服务器与客户端和连接器的兼容性的影响，请参见caching\_sha2\_password作为首选认证插件。**

* + **MySQL现在支持角色，它是命名的权限集合。角色可以被创建和删除。角色可以被授予和撤销权限。角色可以被授予和撤销于用户账户。一个账户的有效适用角色可以从授予该账户的角色中选择，并且可以在该账户的会话中改变。更多信息，见第6.2.10节 "使用角色"。**
  + **MySQL现在包含了用户帐户类别的概念，系统和普通用户根据他们是否有SYSTEM\_USER权限而被区分。见第6.2.11节，"账户类别"。**
  + **以前，除了某些模式外，不可能授予全局性的权限。现在，如果启用了partial\_revokes系统变量，就可以做到这一点。参见第6.2.12节，"使用部分撤销的权限限制"。**
  + **GRANT语句有一个AS user [WITH ROLE]子句，指定了语句执行时要使用的特权上下文的额外信息。这个语法在SQL级别上是可见的，尽管它的主要目的是通过使这些限制出现在二进制日志中，使由部分撤销所施加的授予者特权限制在所有节点上统一复制。见第13.7.1.6节，"GRANT语句"。**
  + **MySQL现在保留了关于密码历史的信息，实现了对以前密码的重复使用的限制。DBA可以要求在一定数量的密码变化或一段时间内，不能从以前的密码中选择新密码。可以在全局以及每个账户的基础上建立密码重复使用策略。**
  + **现在可以要求通过指定要替换的当前密码来验证更改账户密码的尝试。这使DBA能够防止用户在没有证明他们知道当前密码的情况下改变密码。可以在全局和每个账户的基础上建立密码验证策略。**
  + **现在允许账户拥有双重密码，这使得在复杂的多服务器系统中可以无缝地进行分阶段的密码修改，而不需要停机。**
  + **现在，MySQL使管理员能够配置用户账户，从而使由于密码不正确而导致的连续登录失败次数过多会导致账户被临时锁定。每个账户所需的失败次数和锁定时间是可以配置的。**
  + **这些新功能为DBA提供了对密码管理更全面的控制。更多信息请参见第6.2.15节 "密码管理"。**
  + **如果使用OpenSSL进行编译，MySQL现在支持FIPS模式，并且在运行时可以使用OpenSSL库和FIPS对象模块。FIPS模式对加密操作施加了条件，如对可接受的加密算法的限制或对更长的密钥长度的要求。见第6.8节，"FIPS支持"**
  + **现在服务器用于新连接的TLS上下文可以在运行时重新配置。这种能力可能很有用，例如，避免重新启动一个已经运行了很久的MySQL服务器，因为它的SSL证书已经过期。参见服务器端运行时配置和加密连接的监控。**
  + **OpenSSL 1.1.1支持用于加密连接的TLS v1.3协议，MySQL 8.0.16及更高版本也支持TLS v1.3，如果服务器和客户端都使用OpenSSL 1.1.1或更高版本进行编译。参见第6.3.2节，"加密连接TLS协议和密码"。**
  + **MySQL现在将授予命名管道上的客户端的访问控制设置为在Windows上成功通信所需的最小值。较新的MySQL客户端软件可以打开命名管道连接，而无需任何额外的配置。如果旧的客户端软件不能立即升级，新的named\_pipe\_full\_access\_group系统变量可以用来给一个Windows组以必要的权限来打开一个命名的管道连接。完全访问组的成员资格应该是受限制的和临时的。**
* 资源管理。 **MySQL现在支持创建和管理资源组，并允许将服务器内运行的线程分配给特定的组，以便线程根据该组的可用资源执行。组的属性可以控制它的资源，以启用或限制组内线程的资源消耗。DBA可以根据不同的工作负载来修改这些属性。目前，CPU时间是一种可管理的资源，以 "虚拟CPU "的概念为代表，这个术语包括CPU核、超线程、硬件线程等。服务器在启动时确定有多少虚拟CPU可用，具有适当权限的数据库管理员可以将这些CPU与资源组联系起来，并将线程分配给组。欲了解更多信息，请参见第5.1.16节 "资源组"。**
* **表加密管理。 现在可以通过定义和执行加密默认值来全局管理表加密。default\_table\_encryption变量为新创建的模式和一般表空间定义了一个加密默认值。在创建模式时，也可以使用DEFAULT ENCRYPTION子句来定义模式的加密默认值。默认情况下，一个表会继承它所创建的模式或一般表空间的加密。加密默认**值通过启用table\_encryption\_privilege\_check变量来强制执行。当创建或改变具有不同于default\_table\_encryption设置的加密设置的模式或一般表空间时，或者创建或改变具有不同于默认模式加密的加密设置的表时，会发生特权检查。TABLE\_ENCRYPTION\_ADMIN权限允许**在table\_encryption\_privilege\_check被启用时覆盖默认的加密设置。更多信息，请参见为模式和一般表空间定义加密默认值**
* InnoDB的增强功能**。 增加了这些InnoDB的增强功能。**
  + **每次自动增量计数器的值发生变化时，都会将其写入redo log，并在每次检查点时保存到引擎私有的系统表中。这些变化使得当前的最大自动增量计数器的值在服务器重启时是持久的。此外。**
    - **服务器重启不再取消AUTO\_INCREMENT = N表选项的效果。如果你把自动增量计数器初始化为一个特定的值，或者你把自动增量计数器的值改成一个更大的值，新的值会在服务器重新启动时持续存在。**
    - **在ROLLBACK操作之后立即重新启动服务器，不再导致分配给回滚事务的自动增量值的重复使用。**
    - **如果你修改AUTO\_INCREMENT列的值，使其大于当前的最大自动增量值（例如，在UPDATE操作中），新的值会被持久化，随后的INSERT操作会从新的、更大的值开始分配自动增量值。**

**更多信息，请参见第15.6.1.6节，"InnoDB中的AUTO\_INCREMENT处理"，以及InnoDB AUTO\_INCREMENT计数器初始化。**

* **当遇到索引树损坏时，InnoDB会向redo日志写一个损坏标志，这使得损坏标志的崩溃是安全的。InnoDB还在每个检查点上将内存中的corruption标志数据写到引擎私有的系统表中。在恢复过程中，InnoDB从这两个位置读取corruption标志，并在将内存中的表和索引对象标记为corruption之前合并结果。**
* **InnoDB memcached插件支持多个获取操作（在一个memcached查询中获取多个键值对）和范围查询。参见第15.20.4节，"InnoDB memcached多次获取和范围查询支持"。**
* **一个新的动态变量，innodb\_deadlock\_detect，可以用来禁用死锁检测。在高并发系统中，当许多线程等待同一个锁时，死锁检测会导致速度减慢。有时，当死锁发生时，禁用死锁检测并依靠innodb\_lock\_wait\_timeout设置进行事务回滚可能会更有效。**
* **新的INFORMATION\_SCHEMA.INNODB\_CACHED\_INDEXES表报告了每个索引在InnoDB缓冲池中缓存的索引页数量。**
* **InnoDB临时表现在被创建在共享的临时表空间ibtmp1中。**
* **InnoDB表空间加密功能支持对redo log和Undo logs数据进行加密。参见redo log加密，和Undo logs加密。**
* **InnoDB支持NOWAIT和SKIP LOCKED选项与SELECT ... FOR SHARE和SELECT ... FOR UPDATE锁定读取语句。NOWAIT使语句在请求的行被其他事务锁定时立即返回。SKIP LOCKED从结果集中删除锁定的行。参见用NOWAIT和SKIP LOCKED锁定读并发。**
* **SELECT ... FOR SHARE取代了SELECT ... LOCK IN SHARE MODE，但是为了向后兼容，LOCK IN SHARE MODE仍然可用。这些语句是等价的。然而，FOR UPDATE和FOR SHARE支持NOWAIT、SKIP LOCKED和OF tbl\_name选项。参见章节13.2.10, "SELECT语句"。OF tbl\_name将锁定查询应用于命名的表。**
* **ADD PARTITION, DROP PARTITION, COALESCE PARTITION, REORGANIZE PARTITION, 和REBUILD PARTITION ALTER TABLE选项由本地的就地分区API支持，可以和ALGORITHM={COPY|INPLACE}和LOCK子句一起使用。**
* **带有ALGORITHM=INPLACE的DROP PARTITION会删除存储在分区中的数据，并删除该分区。然而，带有ALGORITHM=COPY或old\_alter\_table=ON的DROP PARTITION会重建分区表，并尝试将数据从被删除的分区转移到另一个具有兼容PARTITION ... VALUES定义。不能移动到另一个分区的数据被删除。**
* **InnoDB存储引擎现在使用MySQL数据字典，而不是它自己的特定存储引擎的数据字典。关于数据字典的信息，见第14章，MySQL数据字典。**
* **mysql系统表和数据字典表现在在MySQL数据目录中名为mysql.ibd的单一InnoDB表空间文件中创建。以前，这些表是在mysql数据库目录下的单个InnoDB表空间文件中创建的。**
* **以下是MySQL 8.0中引入的undo tablespaces变化。**
  + **默认情况下，Undo logs现在驻留在两个undo tablespaces中，这些表空间是在MySQL实例初始化时创建的。Undo logs不再在系统表空间中创建。**
  + **从MySQL 8.0.14开始，可以在运行时使用CREATE UNDO TABLESPACE语法在选定的位置创建额外的undo tablespaces。**

CREATE UNDO TABLESPACE ***tablespace\_name*** ADD DATAFILE '***file\_name***.ibu';

**使用CREATE UNDO TABLESPACE语法创建的Undo表空间可以在运行时使用DROP UNDO TABLESPACE语法删除。**

DROP UNDO TABLESPACE ***tablespace\_name***;

**ALTER UNDO TABLESPACE 语法可以用来标记一个undo tablespaces为活动或不活动。**

ALTER UNDO TABLESPACE ***tablespace\_name*** SET {ACTIVE|INACTIVE};

**在 INFORMATION\_SCHEMA.INNODB\_TABLESPACES 表中增加了显示表空间状态的 STATE 列。一个撤销的表空间在被撤销之前必须处于空的状态。**

* **innodb\_undo\_log\_truncate变量默认是启用的。**
* **innodb\_rollback\_segments变量定义了每个undo tablespaces的回滚段的数量。以前，innodb\_rollback\_segments指定了MySQL实例的回滚段的总数。这一变化增加了可用于并发事务的回滚段的数量。更多的回滚段增加了并发事务为Undo logs使用单独的回滚段的可能性，导致更少的资源争夺**
* **修改了影响缓冲池预冲洗和冲洗行为的变量的默认值**
* **innodb\_max\_dirty\_pages\_pct\_lwm的默认值现在是10。之前的默认值为0，禁用缓冲池预冲洗。当缓冲池中的脏页百分比超过10%时，10的值可以启用预冲洗。启用预冲洗可以提高性能的一致性。**
* **innodb\_max\_dirty\_pages\_pct默认值从75增加到90。InnoDB尝试从缓冲池中刷新数据，使脏页的百分比不超过这个值。增加的默认值允许在缓冲池中有更大比例的脏页。**
* **默认的innodb\_autoinc\_lock\_mode设置现在是2（交错）。交错锁模式允许并行地执行多行插入，这提高了并发性和可扩展性。新的innodb\_autoinc\_lock\_mode默认设置反映了MySQL 5.7中从基于语句的复制到基于行的复制作为默认复制类型的变化。基于语句的复制需要连续的自动增量锁模式（以前的默认值），以确保自动增量值以可预测和可重复的顺序分配给给定的SQL语句序列，而基于行的复制对SQL语句的执行顺序不敏感。更多信息，请参阅InnoDB AUTO\_INCREMENT锁模式。**

**对于使用基于语句复制的系统，新的innodb\_autoinc\_lock\_mode默认设置可能会破坏依赖于顺序自动增量值的应用程序。要恢复以前的默认值，请将innodb\_autoinc\_lock\_mode设置为1。**

* **ALTER TABLESPACE ...支持重命名一个普通表空间。语法来支持。**
* **新的innodb\_dedicated\_server变量，默认是禁用的，可以用来让InnoDB根据服务器上检测到的内存量自动配置以下选项**
* **innodb\_buffer\_pool\_size**
* **innodb\_log\_file\_size**
* **innodb\_flush\_method**

**该选项适用于在专用服务器上运行的MySQL服务器实例。欲了解更多信息，请参见第15.8.12节 "为专用的MySQL服务器启用自动配置"。**

* **新的INFORMATION\_SCHEMA.INNODB\_TABLESPACES\_BRIEF视图为InnoDB表空间提供空间、名称、路径、标志和空间类型数据。**
* **与MySQL捆绑的zlib库版本从1.2.3版提高到1.2.11版。MySQL在zlib库的帮助下实现了压缩。**

**如果你使用InnoDB压缩表，请参阅第2.11.4节 "MySQL 8.0中的变化"，了解相关的升级影响。**

* **除了全局临时表空间和undo tablespaces文件外，所有InnoDB表空间文件中都存在序列化字典信息（SDI）。SDI是表和表空间对象的序列化元数据。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，字典对象元数据可以从表空间文件中提取。SDI提取是使用ibd2sdi工具进行的。SDI数据是以JSON格式存储的。**

**在表空间文件中包含SDI数据会增加表空间文件的大小。一个SDI记录需要一个索引页，默认大小为16KB。然而，SDI数据在存储时被压缩，以减少存储空间的占用。**

* **InnoDB存储引擎现在支持原子DDL，这确保了DDL操作被完全提交或回滚，即使服务器在操作过程中停止了。更多信息，请参见第13.1.1节，"原子数据定义语句支持"。**
* **当服务器处于离线状态时，可以使用innodb\_directories选项将表空间文件移动或恢复到一个新的位置。欲了解更多信息，请参见第15.6.3.6节 "在服务器离线时移动表空间文件"。**
* **实现了以下redo log的优化。**
  + **用户线程现在可以同时向日志缓冲区写入，而无需同步写入。**
  + **用户线程现在可以按照宽松的顺序将脏页添加到冲刷列表中。**
  + **一个专门的日志线程现在负责将日志缓冲区写入系统缓冲区，将系统缓冲区刷新到磁盘，通知用户线程关于写入和刷新的重做，维护放松的刷新列表顺序所需的滞后，以及写入检查点。**
  + **增加了系统变量，用于配置等待冲刷重做的用户线程对自旋延迟的使用。**
    - **innodb\_log\_wait\_for\_flush\_spin\_hwm。定义了最大的平均日志刷新时间，超过这个时间，用户线程在等待刷新的重做时不再旋转。**
    - **innodb\_log\_spin\_cpu\_abs\_lwm: 定义最小的CPU使用量，低于该值时，用户线程在等待冲刷的重做时不再旋转。**
    - **innodb\_log\_spin\_cpu\_pct\_hwm: 定义用户线程在等待刷新的重做时不再旋转的最大CPU使用量。**
    - **innodb\_log\_buffer\_size变量现在是动态的，它允许在服务器运行时调整日志缓冲区的大小。**

**欲了解更多信息，请参见第8.5.4节 "优化InnoDBredo log"。**

* **从MySQL 8.0.12开始，Undo logs支持对大对象（LOB）数据的小型更新，这提高了大小为100字节或更小的LOB更新的性能。以前，LOB更新至少要有一个LOB页面大小，这对于可能只修改几个字节的更新来说是不太理想的。这项改进建立在MySQL 8.0.4中对LOB数据的部分更新的支持之上。**
* **从MySQL 8.0.12开始，ALGORITHM=INSTANT被支持用于以下ALTER TABLE操作。**
  + **增加一个列。这个功能也被称为 "即时添加列"。适用的限制条件。参见章节15.12.1, "在线DDL操作"。**
  + **添加或删除一个虚拟列。**
  + **添加或删除一个列的默认值。**
  + **修改一个ENUM或SET列的定义。**
  + **改变索引类型。**
  + **重命名一个表。**

**支持ALGORITHM=INSTANT的操作只修改数据字典中的元数据。在表上没有元数据锁，表的数据不受影响，使得这些操作是瞬时的。如果没有明确指定，ALGORITHM=INSTANT会被支持它的操作默认使用。如果指定了ALGORITHM=INSTANT但不支持，操作会立即失败并出现错误。**

**关于支持ALGORITHM=INSTANT的操作的更多信息，见第15.12.1节 "在线DDL操作"。**

* **从MySQL 8.0.13开始，TempTable存储引擎支持存储二进制大对象（BLOB）类型的列。这一改进提高了使用包含BLOB数据的临时表的查询性能。以前，包含BLOB数据的临时表被存储在由internal\_tmp\_disk\_storage\_engine定义的磁盘存储引擎中。更多信息，见第8.4.4节 "MySQL中的内部临时表使用"。**
* **从MySQL 8.0.13开始，InnoDB静态数据加密功能支持一般表空间。以前，只有文件-每表的表空间可以被加密。为了支持一般表空间的加密，CREATE TABLESPACE和ALTER TABLESPACE语法被扩展到包括一个ENCRYPTION子句。**

**INFORMATION\_SCHEMA.INNODB\_TABLESPACES表现在包括一个ENCRYPTION列，表明表空间是否被加密。**

**增加了 stage/innodb/alter tablespace (encryption) 性能模式阶段工具，以允许监控一般表空间加密操作。**

* **禁用innodb\_buffer\_pool\_in\_core\_file变量，通过排除InnoDB缓冲池页面来减少核心文件的大小。要使用这个变量，core\_file变量必须被启用，并且操作系统必须支持MADV\_DONTDUMP非POSIX扩展到madvise()，在Linux 3.4及以后版本中支持。更多信息，见第15.8.3.7节 "从核心文件中排除缓冲池页"。**
* **从MySQL 8.0.13开始，用户创建的临时表和由优化器创建的内部临时表被存储在会话临时表空间中，这些临时表空间是从临时表空间池中分配给会话的。当一个会话断开连接时，它的临时表空间被截断并释放回池中。在以前的版本中，临时表是在全局临时表空间（ibtmp1）中创建的，在临时表被丢弃后，它不会将磁盘空间返回给操作系统。**

**innodb\_temp\_tablespaces\_dir变量定义了创建会话临时表空间的位置。默认的位置是数据目录下的#innodb\_temp目录。**

**INNODB\_SESSION\_TEMP\_TABLESPACES表提供关于会话临时表空间的元数据。**

**全局临时表空间（ibtmp1）现在存储对用户创建的临时表所做更改的回滚段。**

* **从MySQL 8.0.14开始，InnoDB支持并行集群索引读取，这可以提高CHECK TABLE性能。这个功能不适用于二级索引扫描。innodb\_parallel\_read\_threads会话变量必须被设置为大于1的值，才能发生并行集群索引读取。默认值是4。 用于执行并行集群索引读取的实际线程数由innodb\_parallel\_read\_threads设置或要扫描的索引子树数决定，以较小者为准。**
* **从8.0.14开始，当innodb\_dedicated\_server变量被启用时，日志文件的大小和数量是根据自动配置的缓冲池大小来配置的。以前，日志文件的大小是根据服务器上检测到的内存量来配置的，而日志文件的数量则不是自动配置的。**
* **从8.0.14开始，CREATE TABLESPACE语句的ADD DATAFILE子句是可选的，这允许没有FILE权限的用户创建表空间。在没有ADD DATAFILE子句的情况下执行的CREATE TABLESPACE语句隐含地创建了一个具有唯一文件名的表空间数据文件。**
* **默认情况下，当TempTable存储引擎占用的内存量超过temptable\_max\_ram变量定义的内存限制时，TempTable存储引擎开始从磁盘分配内存映射的临时文件。从MySQL 8.0.16开始，这种行为由temptable\_use\_mmap变量控制。禁用temptable\_use\_mmap导致TempTable存储引擎使用InnoDB磁盘上的内部临时表而不是内存映射的文件作为其溢出机制。欲了解更多信息，请参见内部临时表存储引擎。**
* **从MySQL 8.0.16开始，InnoDB静态数据加密功能支持对mysql系统表空间进行加密。mysql系统表空间包含mysql系统数据库和MySQL数据字典表。欲了解更多信息，请参见第15.13节 "InnoDB静态数据加密"。**
* **在MySQL 8.0.16中引入的innodb\_spin\_wait\_pause\_multiplier变量为自旋锁轮询的持续时间提供了更大的控制，当线程等待获得一个mutex或rw-lock时，会出现自旋锁轮询的延迟。可以对延迟进行更精细的调整，以考虑不同处理器架构上PAUSE指令持续时间的差异。更多信息请参见第15.8.8节，"配置自旋锁轮询"。**
* **在MySQL 8.0.17中，通过更好地利用读取线程，通过减少并行扫描期间发生的预取活动的读取线程I/O，以及通过支持分区的并行扫描，InnoDB的大数据集并行读取线程性能得到了改善。**

**并行读线程功能由innodb\_parallel\_read\_threads变量控制。现在的最大设置是256，这是所有客户端连接的线程总数。如果达到了线程限制，连接会退回到使用单线程。**

* **在MySQL 8.0.18中引入的innodb\_idle\_flush\_pct变量允许在空闲期间对页面刷新进行限制，这可以帮助延长固态存储设备的寿命。见限制闲置期间的缓冲区刷新。**
* **从MySQL 8.0.19开始，支持为生成直方图统计的目的对InnoDB数据进行有效采样。参见直方图统计分析。**
* **从MySQL 8.0.20开始，双写缓冲区的存储区域驻留在双写文件中。在以前的版本中，该存储区位于系统表空间中。将存储区移出系统表空间，可以减少写延迟，增加吞吐量，并为双写缓冲区页面的放置提供灵活性。以下是为高级双写缓冲区配置引入的系统变量。**
  + **innodb\_doublewrite\_dir**

**定义了双写缓冲区的文件目录。**

* **innodb\_doublewrite\_files**

**定义了双写文件的数量。**

* **innodb\_doublewrite\_pages**

**定义每个线程在批量写入时的最大双写页数。**

* **innodb\_doublewrite\_batch\_size**

**定义在一个批次中要写入的双写页数。**

**更多信息，请参见第15.6.4节，"双写缓冲区"。**

* **在MySQL 8.0.20中改进了争夺意识到的事务调度（CATS）算法，该算法优先考虑正在等待锁的事务。事务调度权重的计算现在完全由一个单独的线程执行，这提高了计算性能和准确性。**

**先入先出（FIFO）算法，也被用于事务调度，被删除。FIFO算法由于CATS算法的增强而变得多余了。以前由FIFO算法执行的事务调度现在由CATS算法执行。**

**在 INFORMATION\_SCHEMA.INNODB\_TRX 表中增加了 TRX\_SCHEDULE\_WEIGHT 列，它允许查询 CATS 算法分配的事务调度权重。**

**增加了以下 INNODB\_METRICS 计数器，用于监控代码级的事务调度事件。**

* **lock\_rec\_release\_attempts**

**释放记录锁的尝试次数。**

* **lock\_rec\_grant\_attempts**

**试图授予记录锁的次数。**

* **lock\_schedule\_refreshes**

**分析等待图以更新事务计划权重的次数。**

**欲了解更多信息，请参见第15.7.6节 "事务调度"。**

* **从MySQL 8.0.21开始，为了提高需要访问表和行资源的锁队的操作的并发性，锁系统突变（lock\_sys->mutex）被分片锁所取代，锁队被分组为表和页锁队分片，每个分片由一个专用突变保护。以前，单一锁系统的mutex保护所有的锁队，这在高并发系统中是一个争论点。新的分片实现允许对锁队列进行更细化的访问。**

**锁系统的mutex（lock\_sys->mutex）被以下分片锁所取代。**

* **一个全局锁（lock\_sys->latches.global\_latch）由64个读写锁对象（rw\_lock\_t）组成。对单个锁队的访问需要一个共享的全局锁和一个锁队分片上的锁。需要访问所有锁队列的操作需要一个独占的全局锁，它锁住所有表和页锁队列分片。**
* **表分片锁（lock\_sys->latches.table\_shards.mutexes），实现为一个512个分片的数组，每个分片专门用于512个表锁队列分片之一。**
* **Page shard latches (lock\_sys->latches.page\_shards.mutexes), 实现为512个mutexes的数组，每个mutex专门用于512个page lock queue shard中的一个。**

**性能模式wait/synch/mutex/innodb/lock\_mutex工具用于监控单一锁系统的mutex，被用于监控新的全局、表分片和页分片锁的工具所取代。**

* **wait/synch/sxlock/innodb/lock\_sys\_global\_rw\_lock**
* **wait/synch/mutex/innodb/lock\_sys\_table\_mutex**
* **wait/synch/mutex/innodb/lock\_sys\_page\_mutex**

**从MySQL 8.0.21开始，使用DATA DIRECTORY子句在数据目录之外创建的表和表分区数据文件被限制在InnoDB已知的目录中。这一变化允许数据库管理员控制表空间数据文件的创建位置，并确保在恢复期间可以找到这些数据文件。**

**一般的和每表文件的表空间数据文件（.ibd文件）不能再在undo tablespaces目录（innodb\_undo\_directory）中创建，除非该目录直接被InnoDB知道。**

**已知的目录是那些由datadir、innodb\_data\_home\_dir和innodb\_directories变量定义的目录。**

**Truncating一个驻留在逐个文件表空间中的InnoDB表时，会放弃现有的表空间并创建一个新的表空间。从MySQL 8.0.21开始，InnoDB在默认位置创建新的表空间，如果当前表空间目录未知，则向错误日志写入警告。要让TRUNCATE TABLE在其当前位置创建表空间，在运行TRUNCATE TABLE之前将目录添加到innodb\_directories设置中。**

* **从MySQL 8.0.21开始，可以使用ALTER INSTANCE {ENABLE|DISABLE}启用和禁用redo log。INNODB REDO\_LOG语法启用和禁用。这个功能是为了将数据加载到一个新的MySQL实例。禁用redo log有助于通过避免redo log的写入来加快数据加载速度。**

**新的INNODB\_REDO\_LOG\_ENABLE权限允许启用和禁用redo log记录。**

**新的Innodb\_redo\_log\_enabled状态变量允许监控redo log的状态。**

**参见禁用redo log。**

* **在启动时，InnoDB根据存储在数据字典中的表空间文件路径验证已知的表空间文件的路径，以防表空间文件被移动到不同的位置。新的innodb\_validate\_tablespace\_paths变量在MySQL 8.0.21中引入，允许禁用表空间路径验证。这个功能是为不移动表空间文件的环境准备的。禁用表空间路径验证可以改善有大量表空间文件的系统的启动时间。**

**更多信息，见第15.6.3.7节 "禁用表空间路径验证"。**

* **从MySQL 8.0.21开始，在支持原子DDL的存储引擎上，CREATE TABLE ... SELECT语句在使用基于行的复制时在二进制日志中被记录为一个事务。以前，它被记录为两个事务，一个用于创建表，另一个用于插入数据。有了这个变化，CREATE TABLE ... SELECT 语句现在对基于行的复制是安全的，并且允许与基于 GTID 的复制一起使用。更多信息，请参见第13.1.1节，"原子数据定义语句支持"。**
* **在一个繁忙的系统上截断一个undo tablespaces可能会影响性能，因为相关的冲洗操作会从缓冲池中删除旧的undo tablespaces页面，并将新的undo tablespaces的初始页面冲洗到磁盘。为了解决这个问题，从MySQL 8.0.21开始，冲刷操作被移除。**

**旧的undo tablespaces页在最近使用最少时被动释放，或者在下一个完整检查点时被删除。新的undo tablespaces的初始页现在被重做记录，而不是在截断操作期间被刷入磁盘，这也提高了undo tablespaces截断操作的持久性。**

**为了防止过多的undo tablespaces截断操作引起的潜在问题，在检查点之间对同一undo tablespaces的截断操作现在被限制在64次。如果超过了这个限制，一个undo tablespaces仍然可以变得不活跃，但直到下一个检查点之后才会被截断。**

**删除了与失效的撤消截断冲刷操作相关的INNODB\_METRICS计数器。被删除的计数器包括：undo\_truncate\_sweep\_count、undo\_truncate\_sweep\_usec、undo\_truncate\_flush\_count以及undo\_truncate\_flush\_usec。**

**参见第15.6.3.4节，"undo tablespaces"。**

* **从MySQL 8.0.22开始，新的innodb\_extend\_and\_initialize变量允许配置InnoDB如何分配空间给Linux上的每表文件和一般表空间。默认情况下，当一个操作需要一个表空间的额外空间时，InnoDB会向表空间分配页面，并向这些页面物理写入NULL。如果经常分配新的页面，这种行为会影响性能。你可以在Linux系统上禁用innodb\_extend\_and\_initialize来避免向新分配的表空间页面物理写入NULL。当innodb\_extend\_and\_initialize被禁用时，空间的分配是使用posix\_fallocate()调用，它保留了空间而没有物理写入NULL。**

**posix\_fallocate()操作不是原子性的，这使得在为表空间文件分配空间和更新文件元数据之间有可能发生故障。这样的故障会使新分配的页面处于未初始化的状态，导致InnoDB试图访问这些页面时出现故障。为了防止这种情况，InnoDB在分配一个新的表空间页之前会写一条redo log记录。如果页面分配操作被中断，在恢复过程中会从redo log记录中重放该操作。**

* **从MySQL 8.0.23开始，InnoDB支持对属于加密表空间的双写文件页进行加密。这些页面是使用相关表空间的加密密钥进行加密的。更多信息，请参见第15.13节，"InnoDB静态数据加密"。**
* **temptable\_max\_mmap变量在MySQL 8.0.23中引入，定义了TempTable存储引擎在开始在磁盘上存储内部临时表数据之前允许从内存映射（MMAP）文件分配的最大内存量。设置为0将禁止从MMAP文件中分配。更多信息，见第8.4.4节 "MySQL中的内部临时表使用"。**
* **AUTOEXTEND\_SIZE选项在MySQL 8.0.23中引入，定义了InnoDB在表空间变满时扩展其大小的数量，使得以较大的增量扩展表空间大小成为可能。AUTOEXTEND\_SIZE选项支持CREATE TABLE、ALTER TABLE、CREATE TABLESPACE和ALTER TABLESPACE语句。更多信息，请参见章节15.6.3.9, "表空间AUTOEXTEND\_SIZE配置"。**

**在 INFORMATION\_SCHEMA.INNODB\_TABLESPACES 表中增加了 AUTOEXTEND\_SIZE 大小列。**

* 字符集支持**。 默认的字符集已经从 latin1 改为 utf8mb4。utf8mb4字符集有几个新的排序，包括utf8mb4\_ja\_0900\_as\_cs，这是MySQL中第一个可用于Unicode的日语特定排序。更多信息，见第10.10.1节 "Unicode字符集"**
* JSON的增强**。 对MySQL的JSON功能进行了以下改进或添加。**
* **增加了->>（内联路径）操作符，它相当于在JSON\_EXTRACT()的结果上调用JSON\_UNQUOTE()。**

**这是MySQL 5.7中引入的列路径操作符->的细化；col->>"$.path "等同于JSON\_UNQUOTE(col->"$.path")。内联路径操作符可以在任何可以使用JSON\_UNQUOTE(JSON\_EXTRACT())的地方使用，如SELECT列列表、WHERE和HAVING子句、ORDER BY和GROUP BY子句。更多信息请参见操作符的描述，以及JSON路径语法。**

* **增加了两个JSON聚合函数JSON\_ARRAYAGG()和JSON\_OBJECTAGG()。JSON\_ARRAYAGG()将一个列或表达式作为其参数，并将其结果聚合为一个单一的JSON数组。表达式可以评估为任何MySQL数据类型；这不一定是一个JSON值。JSON\_OBJECTAGG()接收两个列或表达式，它将其解释为一个键和一个值；它将结果作为一个单一的JSON对象返回。更多信息和例子请参见第12.20节 "聚合函数"。**
* **增加了JSON实用函数JSON\_PRETTY()，它以易于阅读的格式输出现有的JSON值；每个JSON对象成员或数组值被打印在单独的一行，子对象或数组相对于其父对象有2个空格。这个函数也适用于可以被解析为JSON值的字符串。**

**更详细的信息和例子，见第12.18.8节，"JSON实用函数"。**

* **当使用ORDER BY对查询中的JSON值进行排序时，每个值现在由排序键的可变长度部分表示，而不是固定的1K大小的部分。在许多情况下，这可以减少过度使用。例如，一个标量的INT甚至BIGINT值实际上只需要很少的字节，所以这个空间的剩余部分（高达90%或更多）被填充物占用了。这一改变对性能有以下好处。**
* **排序缓冲区的空间现在得到了更有效的利用，因此文件排序不需要像固定长度的排序键那样提前或经常刷新到磁盘。这意味着更多的数据可以在内存中进行排序，避免了不必要的磁盘访问。**
* **较短的键可以比长的键更快地进行比较，在性能上有明显的改善。这对于完全在内存中执行的排序以及需要从磁盘写入和读取的排序都是如此。**
* **在MySQL 8.0.2中增加了对JSON列值的部分就地更新的支持，这比完全删除一个现有的JSON值并在其位置上写入一个新的值更有效，就像以前更新任何JSON列时所做的那样。为了应用这种优化，更新必须使用JSON\_SET(), JSON\_REPLACE(), 或JSON\_REMOVE()。新的元素不能被添加到被更新的JSON文档中；文档中的值不能比更新前占用更多空间。请参阅JSON值的部分更新，以了解对这些要求的详细讨论。**

**JSON文档的部分更新可以被写入二进制日志，比记录完整的JSON文档占用更少的空间。当使用基于语句的复制时，部分更新总是这样被记录下来。为了使其在基于行的复制中工作，你必须首先设置binlog\_row\_value\_options=PARTIAL\_JSON；更多信息请参见这个变量的描述。**

* **增加了JSON实用函数JSON\_STORAGE\_SIZE()和JSON\_STORAGE\_FREE()。JSON\_STORAGE\_SIZE()返回在任何部分更新之前用于JSON文档的二进制表示法的存储空间（见前一项），字节数。JSON\_STORAGE\_FREE()显示JSON类型的表列在使用JSON\_SET()或JSON\_REPLACE()进行部分更新后的剩余空间量；如果新值的二进制表示小于先前的值，则该值大于零。**

**这些函数中的每一个也接受JSON文档的有效字符串表示。对于这样的值，JSON\_STORAGE\_SIZE()返回其转换为JSON文档后的二进制表示所使用的空间。对于一个包含JSON文档的字符串表示的变量，JSON\_STORAGE\_FREE()返回0。如果它的(非空)参数不能被解析为有效的JSON文档，这两个函数都会产生一个错误，如果参数是空的，则产生NULL。**

**更多信息和例子，见第12.18.8节，"JSON实用函数"。JSON\_STORAGE\_SIZE()和JSON\_STORAGE\_FREE()已在MySQL 8.0.2中实现。**

* **在MySQL 8.0.2中增加了对XPath表达式中$[1到5]等范围的支持。在这个版本中还增加了对last关键字和相对寻址的支持，例如$[last]总是选择数组中最后一个（编号最高的）元素，而$[last-1]则是倒数第二的元素。last和使用它的表达式也可以包含在范围定义中。例如，$[last-2 to last-1]返回数组中的最后两个元素，但只有一个。参见搜索和修改JSON值，了解更多信息和例子。**
* **增加了一个JSON合并函数，以符合RFC 7396的要求。JSON\_MERGE\_PATCH()，当用于2个JSON对象时，将它们合并成一个单一的JSON对象，其成员是以下集合的联合体。**
* **第一个对象的每个成员，在第二个对象中没有相同键的成员。**
* **第二个对象中的每个成员，在第一个对象中没有具有相同键的成员，并且其值不是JSON null字面。**
* **每个成员的键都存在于两个对象中，并且其在第二个对象中的值不是JSON null字面。**
* **作为这项工作的一部分，JSON\_MERGE()函数被重新命名为JSON\_MERGE\_PRESERVE()。JSON\_MERGE()在MySQL 8.0中继续被认为是JSON\_MERGE\_PRESERVE()的别名，但现在已被废弃，并将在MySQL的未来版本中被删除。**

**更多信息和例子，见第12.18.4节，"修改JSON值的函数"。**

* **实现了重复键的 "最后一个重复键获胜 "的规范化，与RFC 7159和大多数JavaScript解析器一致。这种行为的一个例子显示在这里，其中只有最右边的成员有键x被保留。**

mysql> SELECT **JSON\_OBJECT('x', '32', 'y', '[true, false]',**

> **'x', '"abc"', 'x', '100') AS Result;**

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

| Result |

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

| {"x": "100", "y": "[true, false]"} |

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

1 row in set (0.00 sec)

**插入MySQL JSON列的值也以这种方式进行规范化，如本例所示。**

mysql> **CREATE TABLE t1 (c1 JSON);**

mysql> **INSERT INTO t1 VALUES ('{"x": 17, "x": "red", "x": [3, 5, 7]}');**

mysql> **SELECT c1 FROM t1;**

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

| c1 |

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

| {"x": [3, 5, 7]} |

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

**这是与以前的MySQL版本不兼容的变化，以前的版本在这种情况下使用 "第一个重复的键获胜 "的算法。**

**参见JSON值的规范化、合并和自动包装，了解更多信息和例子。**

* **在MySQL 8.0.4中增加了JSON\_TABLE()函数。该函数接受JSON数据并将其作为具有指定列的关系表返回。**

**这个函数的语法是JSON\_TABLE(expr, path COLUMNS column\_list) [AS] alias)，其中expr是返回JSON数据的表达式，path是应用于源的JSON路径，column\_list是一个列定义的列表。这里显示了一个例子。**

mysql> ***SELECT \****

-> ***FROM***

-> ***JSON\_TABLE(***

-> ***'[{"a":3,"b":"0"},{"a":"3","b":"1"},{"a":2,"b":1},{"a":0},{"b":[1,2]}]',***

-> ***"$[\*]" COLUMNS(***

-> ***rowid FOR ORDINALITY,***

->

-> ***xa INT EXISTS PATH "$.a",***

-> ***xb INT EXISTS PATH "$.b",***

->

-> ***sa VARCHAR(100) PATH "$.a",***

-> ***sb VARCHAR(100) PATH "$.b",***

->

-> ***ja JSON PATH "$.a",***

-> ***jb JSON PATH "$.b"***

-> ***)***

-> ***) AS jt1;***

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

| rowid | xa | xb | sa | sb | ja | jb |

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

| 1 | 1 | 1 | 3 | 0 | 3 | "0" |

| 2 | 1 | 1 | 3 | 1 | "3" | "1" |

| 3 | 1 | 1 | 2 | 1 | 2 | 1 |

| 4 | 1 | 0 | 0 | NULL | 0 | NULL |

| 5 | 0 | 1 | NULL | NULL | NULL | [1, 2] |

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

**JSON源表达式可以是任何产生有效JSON文档的表达式，包括一个JSON字面，一个表列，或一个返回JSON的函数调用，如JSON\_EXTRACT(t1, data, '$.post.comment') 。更多信息请参见第12.18.6节，"JSON表函数"。**

* 数据类型支持**。 MySQL现在支持在数据类型规范中使用表达式作为默认值。这包括使用表达式作为BLOB、TEXT、GEOMETRY和JSON数据类型的缺省值，这些数据类型以前根本不能被分配缺省值。详情请参见第11.6节 "数据类型默认值"。**
* 优化器。 **增加了这些优化器的改进。**
* **MySQL现在支持不可见索引。一个不可见的索引根本不被优化器使用，但在其他方面被正常维护。索引在默认情况下是可见的。不可见的索引使得有可能测试删除索引对查询性能的影响，而不需要进行破坏性的改变，如果索引被证明是必需的，则必须撤销。参见第8.3.12节，"隐形索引"。**
* **MySQL现在支持降序索引。索引定义中的DESC不再被忽略，而是导致按降序存储键值。以前，索引可以按相反的顺序进行扫描，但在性能上会受到影响。降序索引可以按正向顺序扫描，这更有效率。降序索引也使优化器有可能在最有效的扫描顺序中混合了某些列的升序和其他列的降序时使用多列索引。参见第8.3.13节，"降序索引"。**
* **MySQL现在支持创建索引表达式值而不是列值的功能索引关键部分。功能性关键部分可以对无法以其他方式进行索引的值进行索引，例如JSON值。详情请参见第13.1.15节 "CREATE INDEX语句"。**
* **在MySQL 8.0.14及以后的版本中，由常数字面表达式产生的微不足道的WHERE条件在准备期间被删除，而不是在后来的优化期间被删除。在这个过程中提前删除条件使得有外联的查询有可能简化联接，比如这个。**

SELECT \* FROM t1 LEFT JOIN t2 ON ***condition\_1*** WHERE ***condition\_2*** OR 0 = 1

**现在，优化器在准备过程中发现0 = 1总是假的，这使得OR 0 = 1成为多余的，于是将其删除，留下这个。**

SELECT \* FROM t1 LEFT JOIN t2 ON ***condition\_1*** where ***condition\_2***

**现在，优化器可以把这个查询重写成一个内联，像这样。**

SELECT \* FROM t1 LEFT JOIN t2 WHERE ***condition\_1*** AND ***condition\_2***

**更多信息，见第8.2.1.9节 "外连接优化"。**

* **在MySQL 8.0.16及更高版本中，MySQL可以在优化时使用常数折叠来处理列和常数值之间的比较，其中常数超出范围或处于与列的类型有关的范围边界上，而不是在执行时对每行进行比较。例如，给定一个带有TINYINT UNSIGNED列c的表t，优化器可以重写一个条件，比如WHERE c < 256到WHERE 1（并完全优化条件），或者WHERE c >= 255到WHERE c = 255。**

**参见第8.2.1.14节 "恒定折叠优化"，以了解更多信息。**

* **从MySQL 8.0.16开始，用于IN子查询的半连接优化现在也可以应用于EXISTS子查询。此外，优化器现在对附属于子查询的WHERE条件中的琐碎相关的等价谓词进行装饰，以便它们可以与IN子查询中的表达式类似地处理；这适用于EXISTS和IN子查询。**

**欲了解更多信息，请参阅第8.2.2.1节 "用半连接转换优化IN和EXISTS子查询谓词"。**

* **从MySQL 8.0.17开始，服务器在上下文阶段将任何不完整的SQL谓词（即具有WHERE值形式的谓词，其中值是一个列名或常量表达式，并且没有使用比较运算符）内部改写为WHERE值<>0，因此查询解析器、查询优化器和查询执行器只需要使用完整的谓词。**

**这个变化的一个可见效果是，对于布尔值，EXPLAIN输出现在显示的是true和false，而不是1和0。**

**这个变化的另一个影响是，在SQL布尔上下文中对JSON值的评估会执行与JSON整数0的隐式比较。 考虑在这里创建和填充的表。**

mysql> **CREATE TABLE test (id INT, col JSON);**

mysql> **INSERT INTO test VALUES (1, '{"val":true}'), (2, '{"val":false}');**

**以前，当在SQL布尔上下文中进行比较时，服务器试图将提取的真或假值转换为SQL布尔值，如以下使用IS TRUE的查询所示。**

mysql> **SELECT id, col, col->"$.val" FROM test WHERE col->"$.val" IS TRUE;**

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

| id | col | col->"$.val" |

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

| 1 | {"val": true} | true |

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

**在MySQL 8.0.17及以后的版本中，提取的值与JSON整数0的隐式复制会导致不同的结果。**

mysql> **SELECT id, col, col->"$.val" FROM test WHERE col->"$.val" IS TRUE;**

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

| id | col | col->"$.val" |

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

| 1 | {"val": true} | true |

| 2 | {"val": false} | false |

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

**从MySQL 8.0.21开始，你可以对提取的值使用JSON\_VALUE()，在执行测试之前进行类型转换，如图所示。**

mysql> **SELECT id, col, col->"$.val" FROM test**

-> **WHERE JSON\_VALUE(col, "$.val" RETURNING UNSIGNED) IS TRUE;**

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

| id | col | col->"$.val" |

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

| 1 | {"val": true} | true |

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

**同样从MySQL 8.0.21开始，服务器提供了警告 在SQL布尔上下文中评估一个JSON值会对JSON整数0进行隐式比较；如果这不是你想要的，当以这种方式在SQL布尔上下文中比较提取的值时，考虑用JSON\_VALUE RETURNING将JSON转换为一个SQL数字类型。**

* **在MySQL 8.0.17及以后的版本中，具有NOT IN（子查询）或NOT EXISTS（子查询）的WHERE条件在内部被转换为antijoin。(antijoin返回表的所有记录，对于这些记录，在它被连接的表中没有符合连接条件的记录。) 这就删除了子查询，这可以导致更快的查询执行，因为子查询的表现在在顶层被处理。**

**这与现有的外部连接的IS NULL（不存在）优化类似，并重复使用；见EXPLAIN额外信息。**

* **从MySQL 8.0.21开始，单表UPDATE或DELETE语句现在可以在许多情况下使用半联接转换或子查询的具体化。这适用于这里显示的形式的语句。**
  + **UPDATE t1 SET t1.a=value WHERE t1.a IN (SELECT t2.a FROM t2)**
  + **DELETE FROM t1 WHERE t1.a IN (SELECT t2.a FROM t2)**

**对于满足以下条件的单表UPDATE或DELETE，可以这样做。**

* **UPDATE或DELETE语句使用了一个具有[NOT] IN或[NOT] EXISTS谓词的子查询。**
* **该语句没有ORDER BY子句，也没有LIMIT子句。**

**(UPDATE和DELETE的多表版本不支持ORDER BY或LIMIT)。**

* **目标表不支持先读后写删除（仅与NDB表有关）。**
* **根据子查询中包含的任何提示和optimizer\_switch的值，允许半联接或子查询物化。**

**当半连接优化被用于符合条件的单表DELETE或UPDATE时，这在优化器跟踪中是可见的：对于多表语句，跟踪中有一个join\_optimization对象，而对于单表语句没有。这种转换在EXPLAIN FORMAT=TREE或EXPLAIN ANALYZE的输出中也是可见的；单表语句显示<不可由迭代器执行器执行>，而多表语句报告完整计划。**

**从MySQL 8.0.21开始，使用InnoDB表的多表UPDATE语句支持半一致性读取，用于比REPEATABLE READ弱的事务隔离级别。**

* 改进了哈希连接性能**。 MySQL 8.0.23重新实现了用于哈希连接的哈希表，从而使哈希连接性能得到了一些改进。这项工作包括对一个问题的修复（Bug #31516149，Bug #99933），即只有大约2/3分配给连接缓冲区（join\_buffer\_size）的内存可以被哈希连接实际使用。**

**新的哈希表通常比旧的更快，并且在对齐、键/值以及有许多相等键的情况下使用更少的内存。此外，当哈希表的大小增加时，服务器现在可以释放旧内存。**

* 常见的表表达式**。 MySQL现在支持普通表表达式，包括非递归和递归。普通表表达式可以使用命名的临时结果集，通过允许在SELECT语句和某些其他语句前的WITH子句来实现。更多信息，见第13.2.15节 "WITH（普通表表达式）"。**

**从MySQL 8.0.19开始，递归通用表表达式（CTE）的递归SELECT部分支持一个LIMIT子句。也支持带有OFFSET的LIMIT。参见递归公共表表达式，了解更多信息。**

* 窗口函数**。 MySQL现在支持窗口函数，对于查询的每一条记录，使用与该记录相关的记录进行计算。这些包括诸如RANK()、LAG()和NTILE()等函数。此外，现有的几个聚合函数现在可以作为窗口函数使用（例如，SUM()和AVG()）。更多信息，请参见第12.21节，"窗口函数"。**
* 横向派生表**。 一个派生表现在可以在前面加上LATERAL关键字，以指定它被允许在同一个FROM子句中引用（依赖）前面表的列。横向派生表使某些SQL操作成为可能，这些操作不能用非横向派生表来完成，或者需要较低效率的变通方法。参见章节13.2.11.9, "横向派生表"。**
* 单表DELETE语句中的别名**。 在MySQL 8.0.16及更高版本中，单表DELETE语句支持使用表别名。**
* 支持正则表达式**。 以前，MySQL使用Henry Spencer正则表达式库来支持正则表达式运算符（REGEXP、RLIKE）。正则表达式支持已经使用International Components for Unicode（ICU）重新实现，它提供完整的Unicode支持，并且是多字节安全的。REGEXP\_LIKE()函数以REGEXP和RLIKE操作符的方式执行正则表达式匹配，现在是该函数的同义词。此外，REGEXP\_INSTR()、REGEXP\_REPLACE()和REGEXP\_SUBSTR()函数可分别用于查找匹配位置和执行子串替换和提取。regexp\_stack\_limit和regexp\_time\_limit系统变量提供了对匹配引擎资源消耗的控制。更多信息，请参阅第12.8.2节 "正则表达式"。关于使用正则表达式的应用程序可能会受到实现变化的影响的信息，请参阅正则表达式兼容性考虑。**
* 内部临时表**。 TempTable存储引擎取代了MEMORY存储引擎，成为内存中内部临时表的默认引擎。TempTable存储引擎为VARCHAR和VARBINARY列提供有效的存储。internal\_tmp\_mem\_storage\_engine会话变量定义了内存内部临时表的存储引擎。允许的值是TempTable（默认）和MEMORY。temptable\_max\_ram变量定义了TempTable存储引擎在数据被存储到磁盘之前可以使用的最大内存量。**
* 日志记录**。 错误日志被重写以使用MySQL组件架构。传统的错误日志使用内置组件实现，而使用系统日志的日志则作为一个可加载组件实现。此外，还提供了一个可加载的JSON日志写入器。要控制启用哪些日志组件，请使用log\_error\_services系统变量。更多信息，请参阅第5.4.2节 "错误日志"。**
* 备份锁**。 一种新的备份锁允许在线备份期间的DML，同时防止可能导致不一致的快照的操作。新的备份锁由 LOCK INSTANCE FOR BACKUP 和 UNLOCK INSTANCE 语法支持。使用这些语句需要 BACKUP\_ADMIN 权限。**
* 复制**。 对MySQL Replication进行了以下改进。**
  + **MySQL Replication现在支持使用紧凑的二进制格式对JSON文档的部分更新进行二进制记录，比记录完整的JSON文档节省了日志中的空间。当使用基于语句的日志记录时，这种紧凑的日志记录会自动完成，并且可以通过将新的binlog\_row\_value\_options系统变量设置为PARTIAL\_JSON来启用。更多信息，请参阅JSON值的部分更新，以及binlog\_row\_value\_options的描述。**
* 连接管理。 **MySQL服务器现在允许为管理连接专门配置一个TCP/IP端口。这提供了一个替代方案，即在用于普通连接的网络接口上允许单一的管理连接，即使max\_connections连接已经建立。见第5.1.12.1节，"连接接口"。**

**MySQL现在对压缩的使用提供了更多控制，以尽量减少通过连接到服务器发送的字节数。以前，一个给定的连接要么是未压缩的，要么使用zlib压缩算法。现在，它也可以使用zstd算法，并为zstd连接选择一个压缩级别。允许的压缩算法可以在服务器端配置，也可以在客户端程序和参与源/复制或组复制的服务器的连接-起源端配置。更多信息，见第4.2.8节，"连接压缩控制"。**

* 配置。 **整个MySQL的主机名的最大允许长度已经从以前的60个字符的限制提高到255个ASCII字符。这适用于，例如，数据字典、mysql系统模式、性能模式、INFORMATION\_SCHEMA和sys模式中与主机名相关的列；CHANGE MASTER TO语句的MASTER\_HOST值；SHOW PROCESSLIST语句输出中的Host列；账户名中的主机名（如在账户管理语句和DEFINER属性中使用的）；以及与主机名相关的命令选项和系统变量**

注意事项。

* **允许的主机名长度的增加会影响在主机名列上有索引的表。例如，mysql系统模式中索引主机名的表现在有一个明确的ROW\_FORMAT属性DYNAMIC，以适应更长的索引值。**
* **一些文件名值的配置设置可能是基于服务器主机名构建的。允许的值受到底层操作系统的限制，它可能不允许文件名长到包括255个字符的主机名。这影响到 general\_log\_file、log\_error、pid\_file、relay\_log 和 slow\_query\_log\_file 系统变量和相应的选项。如果基于主机名的值对操作系统来说太长，必须提供明确的较短的值。**
* **尽管服务器现在支持255个字符的主机名，但使用--ssl-mode=VERIFY\_IDENTITY选项建立的与服务器的连接受到OpenSSL支持的最大主机名长度的限制。主机名匹配与SSL证书的两个字段有关，它们的最大长度如下。通用名称：最大长度为64；主题替代名称：最大长度为RFC#1034。**
* 插件。 **以前，MySQL插件可以用C或C++编写。插件使用的MySQL头文件现在包含C++代码，这意味着插件必须用C++编写，而不是C。**
* C语言API**。 MySQL C API现在支持异步函数，用于与MySQL服务器的非阻塞通信。每个函数都是现有同步函数的异步对应物。如果从服务器连接的读或写必须等待，同步函数就会阻塞。异步函数使应用程序能够检查服务器连接上的工作是否准备好进行。如果不是，应用程序可以在以后再次检查之前执行其他工作。参见C API异步接口。**
* 用于转换的额外目标类型**。 函数CAST()和CONVERT()现在支持转换为DOUBLE、FLOAT和REAL类型。在MySQL 8.0.17中添加。参见第12.11节，"转换函数和操作符"。**
* JSON模式验证**。 MySQL 8.0.17增加了两个函数JSON\_SCHEMA\_VALID()和JSON\_SCHEMA\_VALIDATION\_REPORT()，用于再次验证JSON模式的JSON文档。JSON\_SCHEMA\_VALID()如果文档对模式进行了验证，则返回TRUE(1)，如果没有则返回FALSE(0)。JSON\_SCHEMA\_VALIDATION\_REPORT() 返回一个包含验证结果详细信息的JSON文档。下面的声明适用于这两个函数。**
  + **模式必须符合JSON模式规范的第4稿。**
  + **支持必需的属性。**
  + **不支持外部资源和$ref关键字。**
  + **支持正则表达式模式；无效的模式会被沉默地忽略。**

**更多信息和例子见12.18.7节 "JSON模式验证函数"。**

* 多值索引。 **从MySQL 8.0.17开始，InnoDB支持创建多值索引，这是一个定义在存储数值数组的JSON列上的二级索引，对于一个数据记录可以有多个索引记录。这样的索引使用一个关键部分的定义，如CAST(data->'$.zipcode' AS UNSIGNED ARRAY)。多值索引被MySQL优化器自动用于合适的查询，可以在EXPLAIN的输出中看到。**

**作为这项工作的一部分，MySQL增加了一个新的函数JSON\_OVERLAPS()和一个新的MEMBER OF()操作符，用于处理JSON文档，另外还用一个新的ARRAY关键字扩展了CAST()函数，如以下列表所述。**

* **JSON\_OVERLAPS()比较了两个JSON文档。如果它们包含任何共同的键值对或数组元素，该函数返回TRUE (1)；否则返回FALSE (0)。如果两个值都是标量，该函数执行一个简单的平等测试。如果一个参数是JSON数组，另一个是标量，标量被视为数组元素。因此，JSON\_OVERLAPS()作为JSON\_CONTAINS()的一个补充。**
* **MEMBER OF()测试第一个操作数（标量或JSON文档）是否是作为第二个操作数传递的JSON数组的成员，如果是，返回TRUE (1)，如果不是，返回FALSE (0)。没有对操作数进行类型转换。**
* **CAST(expression AS type ARRAY)允许通过将json\_path的JSON文档中的JSON数组转换为SQL数组来创建一个功能索引。类型指定器仅限于CAST()已经支持的类型，但BINARY除外（不支持）。CAST()的这种用法（以及ARRAY关键字）只被InnoDB支持，并且只用于创建多值索引。**

**关于多值索引的详细信息，包括例子，参见多值索引。第12.18.3节，"搜索JSON值的函数"，提供了关于JSON\_OVERLAPS()和MEMBER OF()的信息，以及使用的例子。**

* 提示表time\_zone**。 从MySQL 8.0.17开始，time\_zone会话变量可以使用SET\_VAR进行提示。**
* redo log归档**。 从MySQL 8.0.17开始，InnoDB支持redo log归档。当备份操作正在进行时，复制redo log记录的备份工具有时可能无法跟上redo log生成的步伐，导致由于这些记录被覆盖而丢失redo log记录。redo log归档功能通过按顺序将redo log记录写入归档文件来解决这个问题。备份工具可以根据需要从归档文件中复制redo log记录，从而避免了潜在的数据丢失。更多信息，请参阅redo log存档。**
* 克隆插件。 **从MySQL 8.0.17开始，MySQL提供一个克隆插件，允许在本地或从远程MySQL服务器实例克隆InnoDB数据。本地克隆操作在MySQL实例运行的同一服务器或节点上存储克隆的数据。远程克隆操作通过网络将克隆的数据从捐赠者的MySQL服务器实例传输到启动克隆操作的接收者服务器或节点上。**

克隆插件支持复制**。除了克隆数据，克隆操作还从捐赠者那里提取和传输复制坐标，并在接收者那里应用这些坐标，这使得使用克隆插件来配置组复制成员和复制。使用克隆插件来配置，比复制大量的事务要快得多，也更有效率。组复制成员也可以被配置为使用克隆插件作为恢复的替代方法，这样成员就会自动选择最有效的方式来检索种子成员的组数据。**

**更多信息，请参阅第5.6.7节 "克隆插件 "和第18.5.3.2节 "分布式恢复的克隆"。**

* 哈希连接优化**。 从MySQL 8.0.18开始，只要连接中的每一对表至少包括一个等价连接条件，并且没有索引适用于任何连接条件，就会使用散列连接。散列连接不需要索引，尽管它可以使用仅适用于单表谓词的索引。散列连接在大多数情况下比块嵌套循环算法更有效。像这里所示的连接可以用这种方式进行优化。**

SELECT \*

FROM t1

JOIN t2

ON t1.c1=t2.c1;

SELECT \*

FROM t1

JOIN t2

ON (t1.c1 = t2.c1 AND t1.c2 < t2.c2)

JOIN t3

ON (t2.c1 = t3.c1)

**哈希连接也可以用于笛卡尔产品--也就是说，当没有指定连接条件时。**

**你可以使用EXPLAIN FORMAT=TREE或EXPLAIN ANALYZE查看散列连接优化何时被用于一个特定的查询。在MySQL 8.0.20和更高版本中，你也可以使用EXPLAIN，省略FORMAT=TREE）。**

**散列连接可用的内存量受 join\_buffer\_size 值的限制。需要超过这一内存的哈希连接在磁盘上执行；磁盘上哈希连接可以使用的磁盘文件的数量由open\_files\_limit限制。**

**从MySQL 8.0.19开始，不再支持MySQL 8.0.18中引入的hash\_join优化器开关（hash\_join=on仍然作为optimizer\_switch值的一部分出现，但设置它不再有任何影响）。HASH\_JOIN和NO\_HASH\_JOIN优化器提示也不再被支持。开关和提示现在都被废弃了；预计它们将在未来的MySQL版本中被删除。在MySQL 8.0.18及以后的版本中，可以使用NO\_BNL优化器开关禁用哈希连接。**

**在MySQL 8.0.20及以后的版本中，在MySQL服务器中不再使用块嵌套循环，并且在以前使用块嵌套循环的任何时候都会采用散列连接，即使查询不包含等价连接条件。这适用于内部非等价连接、半连接、反连接、左外连接和右外连接。优化器开关（optimizer\_switch）系统变量的block\_nested\_loop标志以及BNL和NO\_BNL优化器提示仍然被支持，但今后只控制使用哈希连接。此外，内部和外部连接（包括半连接和反连接）现在都可以采用分批密钥访问（BKA），它可以逐步分配连接缓冲区的内存，这样单个查询就不需要使用大量的资源来解决它们的实际需求。从MySQL 8.0.18开始，只支持内部连接的BKA。**

**MySQL 8.0.20还用迭代器执行器取代了先前版本的MySQL中使用的执行器。这项工作包括替换旧的索引子查询引擎，这些引擎管辖那些没有被优化为半连接的IN查询的WHERE值IN（SELECT column FROM table WHERE ...）形式的查询，以及以前依赖旧执行器的相同形式的查询。**

**更多信息和例子，请参见章节8.2.1.4，"哈希连接优化"。另请参见Batched Key Access Joins。**

* EXPLAIN ANALYZE 语句。 **在MySQL 8.0.18中实现了EXPLAIN语句的一种新形式，即EXPLAIN ANALYZE，为处理查询时使用的每个迭代器提供TREE格式的SELECT语句执行的扩展信息，并使估计成本与查询的实际成本进行比较成为可能。这些信息包括启动成本、总成本、该迭代器返回的行数，以及执行的循环数。**

**在MySQL 8.0.21及以后的版本中，该语句还支持FORMAT=TREE指定器。TREE是唯一支持的格式。**

**参见用EXPLAIN ANALYZE获取信息，以获得更多信息。**

* 查询投射。 **在8.0.18及以后的版本中，MySQL在参数的数据类型和预期数据类型不匹配的表达式和条件中，将投射操作注入查询项目树。这对查询结果或执行速度没有影响，但使执行的查询等同于符合SQL标准的查询，同时保持与MySQL先前版本的向后兼容性。**

**这种隐式转换现在在时间类型（DATE、DATETIME、TIMESTAMP、TIME）和数字类型（SMALLINT、TINYINT、MEDIUMINT、INT/INTEGER、BIGINT；DECIMAL/NUMERIC； FLOAT、DOUBLE、REAL；BIT）之间进行，只要它们使用任何标准数字比较运算符（=、>=、>、<、<=、<>/！=或<=>）来比较。在这种情况下，任何不属于DOUBLE的值都会被铸成一个。现在还对日期或时间值与DATETIME值之间的比较进行了注入，其中参数在必要时被铸成DATETIME。**

**从MySQL 8.0.21开始，当将字符串类型与其他类型进行比较时，也会执行这种转换。被转换的字符串类型包括CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM和SET。当将一个字符串类型的值与一个数字类型或YEAR进行比较时，字符串被转换为DOUBLE；如果其他参数的类型不是FLOAT、DOUBLE或REAL，它也被转换为DOUBLE。当一个字符串类型与DATETIME或TIMESTAMP值比较时，字符串被转换为DATETIME；当一个字符串类型与DATE比较时，字符串被转换为DATE。**

**通过查看EXPLAIN ANALYZE、EXPLAIN FORMAT=JSON或如这里所示的EXPLAIN FORMAT=TREE的输出，可以看到何时将转换注入给定查询。**

mysql> **CREATE TABLE d (dt DATETIME, d DATE, t TIME);**

Query OK, 0 rows affected (0.62 sec)

mysql> **CREATE TABLE n (i INT, d DECIMAL, f FLOAT, dc DECIMAL);**

Query OK, 0 rows affected (0.51 sec)

mysql> **CREATE TABLE s (c CHAR(25), vc VARCHAR(25),**

-> **bn BINARY(50), vb VARBINARY(50), b BLOB, t TEXT,**

-> **e ENUM('a', 'b', 'c'), se SET('x' ,'y', 'z'));**

Query OK, 0 rows affected (0.50 sec)

mysql> **EXPLAIN FORMAT=TREE SELECT \* from d JOIN n ON d.dt = n.i\G**

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

EXPLAIN: -> Inner hash join *(cast(d.dt as double) = cast(n.i as double))*

(cost=0.70 rows=1)

-> Table scan on n (cost=0.35 rows=1)

-> Hash

-> Table scan on d (cost=0.35 rows=1)

mysql> **EXPLAIN FORMAT=TREE SELECT \* from s JOIN d ON d.dt = s.c\G**

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

EXPLAIN: -> Inner hash join *(d.dt = cast(s.c as datetime(6)))* (cost=0.72 rows=1)

-> Table scan on d (cost=0.37 rows=1)

-> Hash

-> Table scan on s (cost=0.35 rows=1)

1 row in set (0.01 sec)

mysql> **EXPLAIN FORMAT=TREE SELECT \* from n JOIN s ON n.d = s.c\G**

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

EXPLAIN: -> Inner hash join *(cast(n.d as double) = cast(s.c as double))* (cost=0.70 rows=1)

-> Table scan on s (cost=0.35 rows=1)

-> Hash

-> Table scan on n (cost=0.35 rows=1)

1 row in set (0.00 sec)

**通过执行EXPLAIN [FORMAT=TRADITIONAL]也可以看到这种转换，在这种情况下，在执行EXPLAIN语句后还需要发出SHOW WARNINGS。**

* 对TIMESTAMP和DATETIME的时区支持**。 从MySQL 8.0.19开始，服务器接受一个带有插入日期时间（TIMESTAMP和DATETIME）值的时区偏移。这个偏移量使用的格式与设置time\_zone系统变量时使用的格式相同，只是当偏移量的小时部分小于10时需要一个前导零，并且不允许使用'-00:00'。包含时区偏移的日期时间字段的例子是 "2019-12-11 10:40:30-05:00"，"2003-04-14 03:30:00+10:00"，以及 "2020-01-01 15:35:45+05:30"。**

**在选择日期时间值时，不显示时区偏移。**

**包含时区偏移的日期时间字头可以作为准备好的语句参数值使用。**

**作为这项工作的一部分，用于设置time\_zone系统变量的值现在也被限制在-14:00到+14:00之间，包括在内。(只要MySQL时区表被加载，仍有可能将诸如 "EST"、"Posix/Australia/Brisbane "和 "Europe/Stockholm "这样的名称值分配给这个变量；见填充时区表)。**

**更多信息和例子，见第5.1.15节 "MySQL服务器时区支持"，以及第11.2.2节 "DATE、DATETIME和TIMESTAMP类型"。**

* 关于JSON模式CHECK约束失败的精确信息**。 当使用JSON\_SCHEMA\_VALID()指定一个CHECK约束时，MySQL 8.0.19和更高版本提供了关于这种约束失败的原因的精确信息。**

**有关例子和更多信息，请参见JSON\_SCHEMA\_VALID()和CHECK约束。参见章节13.1.20.6, "CHECK 约束"。**

* 具有ON DUPLICATE KEY UPDATE的行和列别名**。 从MySQL 8.0.19开始，可以使用别名来引用要插入的行，以及可选的列。考虑在一个有列a和b的表t上的以下INSERT语句。**

INSERT INTO t SET a=9,b=5

ON DUPLICATE KEY UPDATE a=VALUES(a)+VALUES(b);

**使用新行的别名new，以及在某些情况下，这个行的列的别名m和n，INSERT语句可以用许多不同的方式重写，这里有一些例子。**

INSERT INTO t SET a=9,b=5 AS new

ON DUPLICATE KEY UPDATE a=new.a+new.b;

INSERT INTO t VALUES(9,5) AS new

ON DUPLICATE KEY UPDATE a=new.a+new.b;

INSERT INTO t SET a=9,b=5 AS new(m,n)

ON DUPLICATE KEY UPDATE a=m+n;

INSERT INTO t VALUES(9,5) AS new(m,n)

ON DUPLICATE KEY UPDATE a=m+n;

**更多信息和例子，请参见章节13.2.6.2, "INSERT ... ON DUPLICATE KEY UPDATE语句"。**

* SQL标准显式表子句和表值构造器**。 根据SQL标准，增加了表值构造器和显式表子句。这些在MySQL 8.0.19中实现，分别为TABLE语句和VALUES语句。**

**TABLE语句的格式是TABLE table\_name，相当于SELECT \* FROM table\_name。它支持ORDER BY和LIMIT子句（后者有可选的OFFSET），但不允许选择单个表列。TABLE可以用在任何可以使用等价的SELECT语句的地方；这包括连接、联合、INSERT ... SELECT、REPLACE、CREATE TABLE ... SELECT语句，以及子查询。例如。**

* **TABLE t1 UNION TABLE t2等同于 SELECT \* FROM t1 UNION SELECT \* FROM t2**
* **CREATE TABLE t2 TABLE t1相当于CREATE TABLE t2 SELECT \* FROM t1**
* **SELECT a FROM t1 WHERE b > ANY (TABLE t2)等同于 SELECT a FROM t1 WHERE b > ANY (SELECT \* FROM t2)。**

**VALUES可以用来为INSERT，REPLACE或SELECT语句提供一个表的值，它由VALUES关键字和一系列用逗号分隔的行构造函数（ROW()）组成。例如，语句INSERT INTO t1 VALUES ROW(1,2,3), ROW(4,5,6), ROW(7,8,9)提供了与MySQL特定的INSERT INTO t1 VALUES (1,2,3), (4,5,6), (7,8,9)相一致的SQL兼容性。你也可以像选择表一样从VALUES表值构造器中选择，记住在这样做的时候你必须提供一个表的别名，并且像使用其他任何东西一样使用这个SELECT；这包括连接、联合和子查询。**

**关于TABLE和VALUES的更多信息，以及它们的使用实例，请参见本文档的以下章节。**

* **第13.2.12节，"TABLE语句"**
* **第13.2.14节，"VALUES语句"**
* **第13.1.20.4节，"CREATE TABLE ... SELECT 语句"**
* **第13.2.6.1节 "INSERT ... SELECT 语句"**
* **第13.2.10.2节 "JOIN条款"**
* **第13.2.11节 "子查询"**
* FORCE INDEX、IGNORE INDEX的优化器提示**。 MySQL 8.0引入了索引级优化器提示，作为第8.9.4节 "索引提示 "中描述的传统索引提示的类似物。这里列出了新的提示，以及它们的FORCE INDEX或IGNORE INDEX的对应物。**
  + **GROUP\_INDEX: 等同于FORCE INDEX FOR GROUP BY**
  + **NO\_GROUP\_INDEX：等同于GROUP BY的忽略索引。**
  + **JOIN\_INDEX: 相当于JOIN的FORCE INDEX**

**NO\_JOIN\_INDEX: 相当于忽略JOIN的INDEX**

* + **ORDER\_INDEX：相当于FORCE INDEX FOR ORDER BY。**

**NO\_ORDER\_INDEX：等同于忽略ORDER BY的INDEX。**

* + **INDEX：与GROUP\_INDEX加JOIN\_INDEX加ORDER\_INDEX相同；相当于FORCE INDEX，没有修饰词。**

**NO\_INDEX：与NO\_GROUP\_INDEX加NO\_JOIN\_INDEX加NO\_ORDER\_INDEX相同；相当于没有修饰符的IGNORE INDEX。**

**例如，下面的两个查询是等价的。**

SELECT a FROM t1 FORCE INDEX (i\_a) FOR JOIN WHERE a=1 AND b=2;

SELECT /\*+ JOIN\_INDEX(t1 i\_a) \*/ a FROM t1 WHERE a=1 AND b=2;

**前面列出的优化器提示遵循与现有的索引级优化器提示相同的语法和用法的基本规则。**

**这些优化器提示旨在取代FORCE INDEX和IGNORE INDEX，我们计划在未来的MySQL版本中弃用这两个提示，并随后从MySQL中删除。它们没有实现与USE INDEX完全等同的单一提示；相反，你可以使用NO\_INDEX、NO\_JOIN\_INDEX、NO\_GROUP\_INDEX或NO\_ORDER\_INDEX中的一个或多个来实现同样的效果。**

**更多信息和使用实例，请参见索引级优化器提示。**

* + JSON\_VALUE()函数**。 MySQL 8.0.21实现了一个新函数JSON\_VALUE()，旨在简化JSON列的索引。在其最基本的形式中，它将一个JSON文档和指向该文档中一个单一值的JSON路径作为参数，以及（可选）允许你用RETURNING关键字指定一个返回类型。JSON\_VALUE(json\_doc, path RETURNING type)等同于此。**

CAST(

JSON\_UNQUOTE( JSON\_EXTRACT(***json\_doc***, ***path***) )

AS ***type***

);

**你也可以指定ON EMPTY、ON ERROR或两个子句，类似于JSON\_TABLE()使用的子句。**

**你可以使用JSON\_VALUE()在JSON列的表达式上创建一个索引，像这样。**

CREATE TABLE t1(

j JSON,

INDEX i1 ( (JSON\_VALUE(j, '$.id' RETURNING UNSIGNED)) )

);

INSERT INTO t1 VALUES ROW('{"id": "123", "name": "shoes", "price": "49.95"}');

**使用这个表达式的查询，如这里所示，可以利用索引。**

SELECT name, price FROM t1

WHERE JSON\_VALUE(j, '$.id' RETURNING UNSIGNED) = 123;

**在很多情况下，这比从JSON列创建一个生成的列，然后在生成的列上创建一个索引要简单。**

**更多信息和例子，请看JSON\_VALUE()的描述。**

* + 账户描述和用户属性**。 MySQL 8.0.21引入了在创建或更新用户账户时设置账户描述和用户属性的能力。账户描述由任意的文本组成，作为参数传递给与CREATE USER或ALTER USER语句一起使用的COMMENT子句。用户属性由JSON对象形式的数据组成，作为ATTRIBUTE子句的参数，用于这两个语句中的一个。该属性可以包含任何有效的JSON对象符号的键值对。在一个CREATE USER或ALTER USER语句中只能使用COMMENT或ATTRIBUTE中的一个。**

**账户描述和用户属性在内部作为一个JSON对象一起存储，评论文本作为以评论为键的元素的值。这个信息可以从INFORMATION\_SCHEMA.USER\_ATTRIBUTES表的ATTRIBUTE列中检索；因为它是JSON格式的，你可以使用MySQL的JSON函数和操作符来解析它的内容（见12.18节，"JSON函数"）。对用户属性的连续变化与它的当前值合并，就像使用JSON\_MERGE\_PATCH()函数时一样。**

**例子。**

mysql> **CREATE USER 'mary'@'localhost' COMMENT 'This is Mary Smith\'s account';**

Query OK, 0 rows affected (0.33 sec)

mysql> **ALTER USER 'mary'@'localhost'**

-≫ **ATTRIBUTE '{"fname":"Mary", "lname":"Smith"}';**

Query OK, 0 rows affected (0.14 sec)

mysql> **ALTER USER 'mary'@'localhost'**

-≫ **ATTRIBUTE '{"email":"mary.smith@example.com"}';**

Query OK, 0 rows affected (0.12 sec)

mysql> **SELECT**

-> **USER,**

-> **HOST,**

-> **ATTRIBUTE->>"$.fname" AS 'First Name',**

-> **ATTRIBUTE->>"$.lname" AS 'Last Name',**

-> **ATTRIBUTE->>"$.email" AS 'Email',**

-> **ATTRIBUTE->>"$.comment" AS 'Comment'**

-> **FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='mary' AND HOST='localhost'\G**

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

USER: mary

HOST: localhost

First Name: Mary

Last Name: Smith

Email: mary.smith@example.com

Comment: This is Mary Smith's account

1 row in set (0.00 sec)

**更多信息和例子，请参见第13.7.1.3节 "CREATE USER语句"，第13.7.1.1节 "ALTER USER语句"，以及第26.3.46节 "INFORMATION\_SCHEMA USER\_ATTRIBUTES表"。**

* + 新的optimizer\_switch标志**。 MySQL 8.0.21为optimizer\_switch系统变量增加了两个新的标志，如以下列表中所述。**
    - **prefer\_ordering\_index标志**

**默认情况下，只要优化器确定有LIMIT子句的任何ORDER BY或GROUP BY查询会导致更快的执行，MySQL就会尝试对其使用有序索引。因为在某些情况下，为这种查询选择不同的优化实际上可能表现得更好，现在可以通过将prefer\_ordering\_index标志设置为关闭来禁用这种优化。**

**这个标志的默认值是开。**

* + subquery\_to\_derived标志

**当这个标志被设置为on时，优化器将符合条件的标量子查询转换为派生表的连接。例如，查询SELECT \* FROM t1 WHERE t1.a > (SELECT COUNT(a) FROM t2) 被改写为SELECT t1.a FROM t1 JOIN ( SELECT COUNT(t2.a) AS c FROM t2 ) AS d WHERE t1.a > d.c。**

**这个优化可以应用于一个子查询，它是SELECT, WHERE, JOIN或HAVING子句的一部分；包含一个或多个聚合函数，但没有GROUP BY子句；不是相关的；并且没有使用任何非确定性函数。**

**该优化也可以应用于表子查询，该查询是IN、NOT IN、EXISTS或NOT EXISTS的参数，并且不包含GROUP BY。例如，查询SELECT \* FROM t1 WHERE t1.b < 0 OR t1.a IN (SELECT t2.a + 1 FROM t2) 被改写为SELECT a, b FROM t1 LEFT JOIN (SELECT DISTINCT 1 AS e1, t2.a AS e2 FROM t2) d ON t1.a + 1 = d.e2 WHERE t1.b < 0 OR d.e1 IS NOT NULL。**

**从MySQL 8.0.24开始，这种优化也可以应用于相关的标量子查询，方法是对它应用一个额外的分组，然后对解除的谓词进行外连接。例如，像SELECT \* FROM t1 WHERE (SELECT a FROM t2 WHERE t2.a=t1.a) > 0这样的查询可以改写为SELECT t1.\* FROM t1 LEFT OUTER JOIN (SELECT a, COUNT(\*) AS ct FROM t2 GROUP BY a) AS derived ON t1. MySQL执行了一个cardinality检查，以确保子查询不会返回多于一条记录（ER\_SUBQUERY\_NO\_1\_ROW）。参见第13.2.11.7节 "相关子查询"，以了解更多信息。**

**这个优化通常是禁用的，因为它在大多数情况下不会产生明显的性能优势；这个标志在默认情况下被设置为关闭。**

**更多信息请参见第8.9.2节，"可切换的优化"。另请参见第8.2.1.19节 "LIMIT查询优化"，第8.2.2.1节 "用半连接转换优化IN和EXISTS子查询谓词"，以及第8.2.2.4节 "用合并或物化优化派生表、视图引用和普通表表达式"。**

* + XML的增强**。 从MySQL 8.0.21开始，LOAD XML语句现在支持要导入的XML中的CDATA部分。**
  + 现在支持对YEAR类型的投递**。 从MySQL 8.0.22开始，服务器允许铸造到YEAR。CAST()和CONVERT()函数都支持个位数、两位数和四位数的YEAR值。对于1位数和2位数的值，允许的范围是0-99。四位数的值必须在1901-2155范围内。YEAR也可以作为JSON\_VALUE()函数的返回类型；这个函数只支持四位数的年份。**

**字符串、时间和日期以及浮点值都可以被转换为YEAR。不支持将GEOMETRY值转换为YEAR。**

**更多信息，包括转换规则，见CONVERT()函数的描述。**

* 检索作为UTC的TIMESTAMP值。 **MySQL 8.0.22及更高版本支持在检索时将TIMESTAMP列值从系统时区转换为UTC DATETIME，使用CAST(value AT TIME ZONE specifier AS DATETIME)，其中指定符是[INTERVAL] '+00:00'或'UTC'之一。如果需要的话，可以指定铸模返回的DATETIME值的精度，最多可达小数点后6位。这个结构不支持ARRAY关键字。**

**也支持使用时区偏移将TIMESTAMP值插入到一个表中。对于CONVERT()或任何其他MySQL函数或结构，不支持使用AT TIME ZONE。**

**有关进一步的信息和例子，请参阅CAST()函数的描述。**

* 倾倒文件的输出同步**。 当通过SELECT INTO DUMPFILE和SELECT INTO OUTFILE语句写入文件时，MySQL 8.0.22及更高版本支持定期同步。这可以通过将select\_into\_disk\_sync系统变量设置为ON来启用；写缓冲区的大小由select\_into\_buffer\_size设置的值决定；默认是131072（217）字节。**

**此外，可以使用select\_into\_disk\_sync\_delay来设置同步到磁盘后的可选延迟；默认是没有延迟（0毫秒）。**

**欲了解更多信息，请参见本项目之前提到的变量描述。**

* 语句的单次准备**。 从MySQL 8.0.22开始，准备好的语句只准备一次，而不是每次执行时准备一次。这是在执行PREPARE时进行的。这对于存储过程中的任何语句也是如此；当存储过程第一次执行时，该语句被准备一次。**

**这个变化的一个结果是，在准备好的语句中使用的动态参数的解决方式也以这里列出的方式改变了。**

* **准备好的语句参数在准备语句时被分配了一个数据类型；该类型在语句的每次后续执行中都持续存在（除非语句被重新准备；见下文）。**

**在准备好的语句中为一个给定的参数或用户变量使用不同的数据类型来执行第一次执行后的语句，可能会导致语句被重新准备；由于这个原因，在重新执行准备好的语句时，最好为一个给定的参数使用相同的数据类型。**

* **为了与SQL标准保持一致，以下采用窗口函数的结构不再被接受。**
* **NTILE(NULL)**
* **NTH\_VALUE(expr, NULL)**
* **LEAD(expr, nn)和LAG(expr, nn)，其中nn是一个负数。**

**这有利于更加符合SQL标准。更多细节请参见各个函数的描述。**

* **现在，在准备好的语句中引用的用户变量，其数据类型在语句准备时就已确定；该类型在语句的每次后续执行中都会持续存在。**
* **在一个存储过程中出现的语句所引用的用户变量现在在第一次执行该语句时确定其数据类型；该类型在随后的任何包含存储过程的调用中持续存在。**
* **当执行形式为SELECT expr1, expr2, ...的准备好的语句时。FROM表ORDER BY ? 时，为参数传递一个整数值N不再导致按选择列表中的第N个表达式对结果进行排序；结果不再被排序，正如对ORDER BY常数的预期。**

**对作为准备好的语句或在存储过程中使用的语句只准备一次，可以提高该语句的性能，因为它否定了重复准备的附加成本。这样做还可以避免准备结构可能的多次回滚，这一直是MySQL中许多问题的来源。**

**更多信息，见第13.5.1节 "PREPARE语句"。**

* RIGHT JOIN作为LEFT JOIN处理**。 从MySQL 8.0.22开始，服务器在内部将RIGHT JOIN的所有实例处理为LEFT JOIN，消除了一些在解析时没有进行完全转换的特殊情况。**
* 派生条件推倒优化。 **MySQL 8.0.22（及更高版本）为具有物化派生表的查询实现了派生条件推倒。对于诸如SELECT \* FROM (SELECT i, j FROM t1) AS dt WHERE i > constant这样的查询，现在在许多情况下可以将外层的WHERE条件下推到派生表，在这种情况下会产生SELECT \* FROM (SELECT i, j FROM t1 WHERE i > constant) AS dt。**

**以前，如果派生表被物化而没有合并，MySQL会物化整个表，然后用WHERE条件限定行。使用派生条件推倒优化将WHERE条件移到子查询中，通常可以减少必须处理的行的数量，这可以减少执行查询所需的时间。**

**当派生表不使用任何聚合或窗口函数时，一个外部的WHERE条件可以直接下推到一个物化派生表。当派生表有一个GROUP BY并且不使用任何窗口函数时，外部WHERE条件可以作为一个HAVING条件被推送到派生表。当派生表使用窗口函数，并且外部WHERE引用窗口函数的PARTITION子句中使用的列时，WHERE条件也可以向下推送。**

**派生条件推倒是默认启用的，如optimizer\_switch系统变量的派生\_条件\_推倒标志所指示。该标志是在MySQL 8.0.22中添加的，默认情况下被设置为打开；要为特定查询禁用优化，可以使用NO\_DERIVED\_CONDITION\_PUSHDOWN优化器提示（也在MySQL 8.0.22中添加）。如果由于派生条件推倒被设置为关闭而导致优化被禁用，你可以使用DERIVED\_CONDITION\_PUSHDOWN为特定查询启用它。**

**派生条件推倒优化不能用于包含UNION或LIMIT子句的派生表。此外，一个本身使用子查询的条件不能被推倒，一个WHERE条件不能被推倒到也是外连接的内表的派生表上。关于其他的信息和例子，请看第8.2.2.5节，"派生条件推倒优化"**

* 对MySQL授予表的非锁定读取。 **从MySQL 8.0.22开始，为了允许在MySQL授权表上进行并发的DML和DDL操作，以前在MySQL授权表上获得行锁的读取操作被作为非锁定读取执行。**

**现在在MySQL授予表上作为非锁定读取执行的操作包括。**

* **通过连接列表和子查询从授予表读取数据的SELECT语句和其他只读语句，包括SELECT ... FOR SHARE语句，使用任何事务隔离级别。**
* **使用任何事务隔离级别，从授予表（通过连接列表或子查询）读取数据但不修改数据的DML操作。**

**有关其他信息，请参见授予表的并发性。**

**在MySQL 8.0中废弃的功能**

以下功能在MySQL 8.0中被废弃，并可能在未来的系列中被删除。在显示替代功能的地方，应更新应用程序以使用它们。

对于使用在MySQL 8.0中被废弃的功能的应用程序，如果在更高的MySQL系列中被删除，当从MySQL 8.0源复制到更高系列的副本时，语句可能会失败，或者可能对源和副本产生不同的影响。为了避免这类问题，应修改使用在8.0中被废弃的功能的应用程序，以避免它们，并尽可能使用替代方案。

* utf8mb3字符集已被废弃。请使用utf8mb4代替。
* 由于caching\_sha2\_password是MySQL 8.0中的默认认证插件，并提供了sha256\_password认证插件的功能超集，因此sha256\_password已被废弃；预计它将在MySQL的未来版本中被移除。使用sha256\_password认证的MySQL账户应迁移到使用caching\_sha2\_password来代替。
* validate\_password插件已被重新实现，以使用组件基础设施。validate\_password的插件形式仍然可用，但现在已被废弃；预计它将在未来的MySQL版本中被删除。使用该插件的MySQL安装应过渡到使用组件。见第6.4.3.3节，"过渡到密码验证组件"
* ALTER TABLESPACE和DROP TABLESPACE语句的ENGINE子句已被废弃。
* PAD\_CHAR\_TO\_FULL\_LENGTH SQL模式已被废弃。
* 对于FLOAT和DOUBLE类型的列（以及任何同义词）的AUTO\_INCREMENT支持已被废弃。考虑从这些列中移除AUTO\_INCREMENT属性，或者将其转换为整数类型。
* 对于FLOAT、DOUBLE和DECIMAL类型的列（以及任何同义词），UNSIGNED属性已被废弃。对于这样的列，可以考虑使用一个简单的CHECK约束来代替。
* FLOAT(M,D)和DOUBLE(M,D)语法用于为FLOAT和DOUBLE类型的列（以及任何同义词）指定数字数量，是一个非标准的MySQL扩展。这种语法已被废弃。
* 对于数字数据类型，ZEROFILL属性已被废弃，整数数据类型的显示宽度属性也被废弃。考虑使用其他方法来产生这些属性的效果。例如，应用程序可以使用LPAD()函数将数字零填充到所需的宽度，或者他们可以将格式化的数字存储在CHAR列。
* 对于字符串数据类型，BINARY属性是一个非标准的MySQL扩展，它是指定列字符集（如果没有指定列字符集，则是表默认字符集）的二进制（\_bin）整理的速记。在MySQL 8.0中，BINARY的这种非标准使用是不明确的，因为utf8mb4字符集有多种\_bin校对，所以BINARY属性已被废弃；预计在MySQL的未来版本中会删除对它的支持。应用程序应调整为使用明确的\_bin校对。
* 使用BINARY来指定数据类型或字符集的做法保持不变。
* 非标准的C风格的&&、||和！运算符分别是标准SQL的AND、OR和NOT运算符的同义词，已被废弃。使用非标准操作符的应用程序应该被调整为使用标准操作符。

注意

除非启用了PIPES\_AS\_CONCAT SQL模式，否则||的使用已被弃用。在这种情况下，||标志着SQL标准的字符串连接操作符）。)

* JSON\_MERGE()函数已被废弃。请使用JSON\_MERGE\_PRESERVE()代替。
* SQL\_CALC\_FOUND\_ROWS查询修饰符和附带的FOUND\_ROWS()函数已被弃用。关于替代策略的信息，请参见FOUND\_ROWS()描述。
* 从MySQL 8.0.13开始，对TABLESPACE = innodb\_file\_per\_table和TABLESPACE = innodb\_temporary子句与CREATE TEMPORARY TABLE的支持已被弃用。
* 对于SELECT语句，从MySQL 8.0.20开始，在FROM后面而不是在SELECT的末尾使用INTO子句已被弃用。最好将INTO放在语句的末尾。
* 对于UNION语句，从MySQL 8.0.20开始，包含INTO的这两个变体被弃用。
* 在一个查询表达式的尾部查询块中，在FROM之前使用INTO。
* 在查询表达式的尾部括号内，使用INTO，无论其相对于FROM的位置如何。参见章节13.2.10.1, "SELECT ... INTO语句 "和章节13.2.10.3, "UNION语句"。
* 从MySQL 8.0.23开始，FLUSH HOSTS被弃用。相反，截断性能模式host\_cache表。

TRUNCATE TABLE performance\_schema.host\_cache;

TRUNCATE TABLE操作需要该表的DROP权限。

* mysql\_upgrade客户端已被废弃，因为其升级mysql系统模式中的系统表和其他模式中的对象的功能已被移到MySQL服务器中。参见第2.11.3节，"MySQL升级过程升级的内容"
* --no-dd-upgrade服务器选项已被弃用。它被--upgrade选项所取代，后者对数据字典和服务器升级行为提供了更精细的控制。
* mysql\_upgrade\_info文件是创建数据目录并用于存储MySQL版本号的，已被废弃；预计它将在MySQL的未来版本中被删除。
* relay\_log\_info\_file系统变量和-master-info-file选项已被弃用。以前，当relay\_log\_info\_repository=FILE和master\_info\_repository=FILE被设置时，这些被用来指定中继日志信息日志和源信息日志的名称，但这些设置已被弃用。为中继日志信息日志和源信息日志使用文件已经被崩溃安全的复制表所取代，在MySQL 8.0中是默认的。
* max\_length\_for\_sort\_data系统变量现在已被废弃，因为优化器的变化使其过时且无影响。
* 这些用于压缩与服务器的连接的传统参数已被废弃。--compress客户端命令行选项；mysql\_options()C API函数的MYSQL\_OPT\_COMPRESS选项；slave\_compressed\_protocol系统变量。关于要使用的参数的信息，见第4.2.8节，"连接压缩控制"。
* 使用MYSQL\_PWD环境变量来指定MySQL密码的做法已被弃用。
* 在INSERT ... ON DUPLICATE KEY UPDATE中使用VALUES()来访问新行的值，从MySQL 8.0.20开始已被废弃。使用新行和列的别名来代替。
* 因为在调用JSON\_TABLE()时，在ON EMPTY之前指定ON ERROR是违反SQL标准的，这种语法现在在MySQL中被废弃。从MySQL 8.0.20开始，只要你试图这样做，服务器就会打印出一个警告。当在单个JSON\_TABLE()调用中指定这两个子句时，确保首先使用ON EMPTY。
* 带有索引前缀的列从未被支持为表的分区键的一部分；以前，在创建、改变或升级分区表时，这些列是被允许的，但被表的分区函数排除在外，而且服务器不会发出发生这种情况的警告。这种允许的行为现在已被废弃，并将在未来的MySQL版本中被删除，在分区键中使用任何这样的列会导致它们发生的CREATE TABLE或ALTER TABLE语句被拒绝。

从MySQL 8.0.21开始，只要使用索引前缀的列被指定为分区键的一部分，就会为每个这样的列产生一个警告。每当CREATE TABLE或ALTER TABLE语句因为提议的分区键中的所有列都有索引前缀而被拒绝时，产生的错误现在提供了拒绝的确切原因。在这两种情况下，这包括在分区函数中使用的列通过采用空的PARTITION BY KEY()子句被隐含地定义为表的主键的情况。

更多信息和例子，请看列索引前缀不支持键分区。

* 从MySQL 8.0.22开始，InnoDB memcached插件已被废弃；预计在未来的MySQL版本中会移除对它的支持。
* 从MySQL 8.0.25开始，temptable\_use\_mmap变量已被废弃；预计在未来的MySQL版本中会移除对它的支持。

**在MySQL 8.0中删除的功能**

以下项目已经过时，并在MySQL 8.0中被删除。在显示替代方案的地方，应用程序应更新以使用它们。

对于使用MySQL 8.0中删除的特性的MySQL 5.7应用程序，当从MySQL 5.7源复制到MySQL 8.0副本时，语句可能会失败，或者可能对源和副本产生不同的影响。为避免此类问题，应修改使用MySQL 8.0中删除的特性的应用程序，以避免它们，并尽可能使用替代方案。

* innodb\_locks\_unsafe\_for\_binlog系统变量被删除。READ COMMITTED隔离级别提供了类似的功能。
* 在MySQL 8.0.0中引入的information\_schema\_stats变量被删除，并在MySQL 8.0.3中被information\_schema\_stats\_expiry取代。
* information\_schema\_stats\_expiry定义了缓存的INFORMATION\_SCHEMA表统计数据的过期设置。更多信息，见第8.2.3节，"优化INFORMATION\_SCHEMA查询"。
* 与过时的InnoDB系统表有关的代码在MySQL 8.0.3中被删除。基于InnoDB系统表的INFORMATION\_SCHEMA视图被数据字典表的内部系统视图取代。受影响的InnoDB INFORMATION\_SCHEMA视图被重新命名。

**Table1.1 Renamed InnoDB Information Schema Views**

|  |  |
| --- | --- |
| **Old Name** | **New Name** |
| **INNODB\_SYS\_COLUMNS** | **INNODB\_COLUMNS** |
| **INNODB\_SYS\_DATAFILES** | **INNODB\_DATAFILES** |
| **INNODB\_SYS\_FIELDS** | **INNODB\_FIELDS** |
| **INNODB\_SYS\_FOREIGN** | **INNODB\_FOREIGN** |
| **INNODB\_SYS\_FOREIGN\_COLS** | **INNODB\_FOREIGN\_COLS** |
| **INNODB\_SYS\_INDEXES** | **INNODB\_INDEXES** |
| **INNODB\_SYS\_TABLES** | **INNODB\_TABLES** |
| **INNODB\_SYS\_TABLESPACES** | **INNODB\_TABLESPACES** |
| **INNODB\_SYS\_TABLESTATS** | **INNODB\_TABLESTATS** |
| **INNODB\_SYS\_VIRTUAL** | **INNODB\_VIRTUAL** |

在升级到MySQL 8.0.3或更高版本后，更新任何引用以前InnoDB INFORMATION\_SCHEMA视图名称的脚本。

* 以下与账户管理有关的功能被删除。
  + 使用GRANT来创建用户。相反，使用CREATE USER。遵循这种做法使得NO\_AUTO\_CREATE\_USER SQL模式对GRANT语句不重要，所以它也被删除了，并且当选项文件中sql\_mode选项的这个值阻止mysqld启动时，现在会有一个错误被写入服务器日志。
  + 使用GRANT来修改权限分配以外的账户属性。这包括认证、SSL和资源限制属性。相反，在创建账户时用CREATE USER建立这些属性，或在之后用ALTER USER修改它们
  + CREATE USER和GRANT的IDENTIFIED BY PASSWORD 'auth\_string' 语法。相反，在CREATE USER和ALTER USER中使用IDENTIFIED WITH auth\_plugin AS 'auth\_string'，其中'auth\_string'的值是与命名的插件兼容的格式。
  + 此外，由于IDENTIFIED BY PASSWORD语法被删除，log\_builtin\_as\_identified\_by\_password系统变量是多余的，被删除。
  + PASSWORD()函数。此外，PASSWORD()的移除意味着SET PASSWORD ... = PASSWORD('auth\_string')语法不再可用。
  + old\_passwords系统变量。
* 查询缓存被移除。移除包括这些项目。
  + FLUSH QUERY CACHE和RESET QUERY CACHE语句。
  + 这些系统变量：query\_cache\_limit, query\_cache\_min\_res\_unit, query\_cache\_size, query\_cache\_type, query\_cache\_wlock\_invalidate。
  + 这些状态变量。Qcache\_free\_blocks, Qcache\_free\_memory, Qcache\_hits, Qcache\_inserts, Qcache\_lowmem\_prunes, Qcache\_not\_cached, Qcache\_queries\_in\_cache, Qcache\_total\_blocks。
  + 这些线程状态：检查缓存查询的权限，检查查询缓存的查询，无效查询缓存条目，发送缓存结果到客户端，将结果存储在查询缓存中，等待查询缓存锁。
  + SQL\_CACHE SELECT修改器。

这些被废弃的查询缓存项目仍然被废弃，但没有影响；预计它们将在未来的MySQL版本中被移除。

* SQL\_NO\_CACHE SELECT修改器。
* ndb\_cache\_check\_time系统变量。

have\_query\_cache系统变量仍然是废弃的，并且总是有一个NO的值；预计它将在未来的MySQL版本中被删除。

* 数据字典提供了关于数据库对象的信息，因此服务器不再检查数据目录中的目录名来寻找数据库。因此，--ignore-db-dir选项和ignore\_db\_dirs系统变量是多余的，并被删除。
* DDL日志，也被称为元数据日志，已被删除。从MySQL 8.0.3开始，该功能由数据字典innodb\_ddl\_log表处理。参见查看DDL日志。
* tx\_isolation和tx\_read\_only系统变量已被删除。请使用 transaction\_isolation 和 transaction\_read\_only 来代替。
* sync\_frm 系统变量已经被删除， 因为 .frm 文件已经过时。
* 移除了 secure\_auth 系统变量和 --secure-auth 客户端选项。mysql\_options() C API函数的MYSQL\_SECURE\_AUTH选项已被删除。
* 删除了multi\_range\_count系统变量。
* 移除了log\_warnings系统变量和-log-warnings服务器选项。使用 log\_error\_verbosity 系统变量来代替。
* 删除了 sql\_log\_bin 系统变量的全局范围。 sql\_log\_bin 只有会话范围，依赖访问 @@GLOBAL.sql\_log\_bin 的应用程序应该被调整。
* 删除了 metadata\_locks\_cache\_size 和 metadata\_locks\_hash\_instances 系统变量。
* 删除了未使用的 date\_format、datetime\_format、time\_format 和 max\_tmp\_tables 系统变量。
* 删去了这些已经废弃的兼容SQL模式。DB2, maxdb, mssql, mysql323, mysql40, oracle, postgresql, no\_field\_options, no\_key\_options, no\_table\_options。它们不能再被分配给sql\_mode系统变量或作为mysqldump --兼容选项的允许值。
* 移除MAXDB意味着CREATE TABLE或ALTER TABLE的TIMESTAMP数据类型被视为TIMESTAMP，而不再被视为DATETIME。
* 删除了GROUP BY子句中被废弃的ASC或DESC限定符。以前依赖GROUP BY排序的查询可能产生与以前的MySQL版本不同的结果。为了产生一个给定的排序顺序，提供一个ORDER BY子句。
* EXPLAIN语句的EXTENDED和PARTITIONS关键字已被删除。这些关键字是不必要的，因为它们的效果总是被启用。
* 删除了这些与加密有关的项目。
  + ENCODE()和DECODE()函数。
  + ENCRYPT()函数。
  + DES\_ENCRYPT()和DES\_DECRYPT()函数，--des-key-file选项，have\_crypt系统变量，FLUSH语句的DES\_KEY\_FILE选项，以及HAVE\_CRYPT CMake选项。
* **来代替被删除的加密函数。对于ENCRYPT()，考虑使用SHA2()来代替单向散列。对于其他的，考虑使用AES\_ENCRYPT()和AES\_DECRYPT()代替。**
* **在MySQL 5.7中，多个名称下的空间函数被废弃，以使空间函数名称空间更加一致，目标是每个空间函数名称以ST\_开头，如果它执行精确操作，或以MBR开头，如果它执行基于最小边界矩形的操作。在MySQL 8.0中，被废弃的函数被删除，只留下相应的ST\_和MBR函数。**
  + **这些函数被删除，以支持MBR的名称。Contains(), Disjoint(), Equals(), Intersects(), Overlaps(), Within()。**
  + **这些函数被删除，改用ST\_名称。Area(), AsBinary(), AsText(), AsWKB(), AsWKT(), Buffer(), Centroid(), ConvexHull(), Crosses(), Dimension(), Distance(), EndPoint(), Envelope(), ExteriorRing(), GeomCollFromText(), GeomCollFromWKB(), GeomFromText(), GeomFromWKB() 。GeometryCollectionFromText(), GeometryCollectionFromWKB(), GeometryFromText(), GeometryFromWKB(), GeometryN(), GeometryType(), InteriorRingN(), IsClosed(), IsEmpty(), IsSimple(), LineFromText(), LineFromWKB(), LineStringFromText() 。LineStringFromWKB(), MLineFromText(), MLineFromWKB(), MPPointFromText(), MPPointFromWKB(), MPolyFromText(), MPolyFromWKB(), MultiLineStringFromText(), MultiLineStringFromWKB(), MultiPointFromText(), MultiPointFromWKB(), MultiPolygonFromText() 。MultiPolygonFromWKB(), NumGeometries(), NumInteriorRings(), NumPoints(), PointFromText(), PointFromWKB(), PointN(), PolyFromText(), PolyFromWKB(), PolygonFromText(), PolygonFromWKB(), SRID(), StartPoint(), Touches(), X(), Y()。**
  + **移除GLength()，改用ST\_Length()。**
* **第12.17.4节 "从WKB值创建几何值的函数 "中描述的函数以前接受WKB字符串或几何参数。几何参数不再被允许，并产生一个错误。请参阅该节，了解不再使用几何参数的查询指南。**
* **在SQL语句中，解析器不再将 \N 作为NULL的同义词。使用NULL代替。**

**这个变化不影响用LOAD DATA或SELECT ... INTO OUTFILE进行的文本文件导入或导出操作，对于这些操作，NULL继续用 \N 表示。参见第13.2.7节，"LOAD DATA语句"。**

* **PROCEDURE ANALYSE()语法被删除。**
* **删除了客户端的 --ssl 和 --ssl-verify-server-cert 选项。使用 --ssl-mode=REQUIRED 而不是 --ssl=1 或 --enable-ssl。使用 --ssl-mode=DISABLED 来代替 --ssl=0, --skip-ssl, 或 --disable-ssl。使用 --ssl-mode=VERIFY\_IDENTITY 而不是 --ssl-verify-server-cert 选项。(服务器端的--ssl选项仍然可用，但从MySQL 8.0.25开始已被废弃，并将在未来的MySQL版本中被删除)。**

**对于C API，mysql\_options()的MYSQL\_OPT\_SSL\_ENFORCE和MYSQL\_OPT\_SSL\_VERIFY\_SERVER\_CERT选项对应于客户端的--ssl和--ssl-verify-server-cert选项，并被删除。使用MYSQL\_OPT\_SSL\_MODE，选项值为SSL\_MODE\_REQUIRED或SSL\_MODE\_VERIFY\_IDENTITY来代替。**

* **删除了--temp-pool服务器选项。**
* **移除ignore\_builtin\_innodb系统变量。**
* **服务器不再执行将包含特殊字符的前MySQL 5.1数据库名称转换为添加了#mysql50#前缀的5.1格式。由于不再执行这些转换，mysqlcheck的--fix-db-names和--fix-table-names选项、ALTER DATABASE语句的UPGRADE DATA DIRECTORY NAME子句和Com\_alter\_db\_upgrade状态变量被删除。**
* **只支持从一个主要版本升级到另一个主要版本（例如，从5.0到5.1，或从5.1到5.5），所以应该没有什么剩余的需要将旧的5.0数据库名称转换到当前版本的MySQL。作为一种变通方法，在升级到较新的版本之前，将MySQL 5.0安装升级到MySQL 5.1。**
* **mysql\_install\_db程序已从MySQL发行版中删除。数据目录的初始化应该通过调用mysqld的-initialize或-initialize-insecure选项来进行，而不是通过调用mysqld。此外，mysql\_install\_db使用的mysqld的--bootstrap选项被移除，控制mysql\_install\_db安装位置的INSTALL\_SCRIPTDIR CMake选项被移除。**
* **从MySQL服务器中删除了通用分区处理程序。为了支持给定表的分区，用于该表的存储引擎现在必须提供它自己的（"本地"）分区处理程序。从MySQL服务器中删除了--partition和--skip-partition选项，并且在SHOW PLUGINS的输出或INFORMATION\_SCHEMA.PLUGINS表中不再显示与分区相关的条目。**

**目前有两个MySQL存储引擎提供本地分区支持。InnoDB和NDB。其中，只有InnoDB在MySQL 8.0中被支持。在MySQL 8.0中使用任何其他存储引擎创建分区表的尝试都会失败**

**对升级的影响。** 不支持使用InnoDB以外的存储引擎（如MyISAM）将分区表从MySQL 5.7（或更早）直接升级到MySQL 8.0。处理这样的表有两个选择。

* 删除表的分区，使用ALTER TABLE ... 删除分区
* **将表使用的存储引擎改为InnoDB，使用ALTER TABLE ... ENGINE=INNODB。**

**在将服务器升级到MySQL 8.0之前，必须对每个分区的非InnoDB表执行刚才列出的两个操作中的至少一个。否则，这样的表在升级后不能使用。**

由于会导致使用不支持分区的存储引擎的分区表的表创建语句现在以错误（ER\_CHECK\_NOT\_IMPLEMENTED）失败，你必须确保你希望导入到MySQL 8.0服务器的旧版本的MySQL的转储文件（例如由mysqldump编写的）中创建分区表的任何语句不会指定诸如MyISAM这样没有本地分区处理器的存储引擎。你可以通过执行以下任何一项来做到这一点。

* **从CREATE TABLE语句中删除对分区的任何引用，这些语句使用的STORAGE ENGINE选项的值不是InnoDB。**
* **将存储引擎指定为InnoDB，或者允许InnoDB被默认为表的存储引擎。**

**更多信息请参见第24.6.2节，"与存储引擎有关的分区限制"。**

* 在 INFORMATION\_SCHEMA 中不再维护系统和状态变量信息。这些表被移除。global\_variables, session\_variables, global\_status, session\_status。使用相应的性能模式表来代替。参见第27.12.14节 "性能模式系统变量表 "和第27.12.15节 "性能模式状态变量表"。此外， show\_compatibility\_56 系统变量被删除了。在过渡时期，INFORMATION\_SCHEMA表中的系统和状态变量信息被转移到Performance Schema表**中**，因此不再需要它。这些状态变量被移除。Slave\_heartbeat\_period、Slave\_last\_heartbeat、Slave\_received\_heartbeats、Slave\_retried\_transactions、Slave\_running。它们提供的信息可以在Performance Schema表中找到；请看迁移到Performance Schema系统和状态变量表。
* **性能模式的 setup\_timers 表被删除了，performance\_timers 表中的 TICK 行也被删除了。**
* **libmysqld嵌入式服务器库被移除，同时也被移除。**
  + **mysql\_options() MYSQL\_OPT\_GUESS\_CONNECTION、MYSQL\_OPT\_USE\_EMBEDDED\_CONNECTION、MYSQL\_OPT\_USE\_REMOTE\_CONNECTION和MYSQL\_SET\_CLIENT\_IP选项**
  + **mysql\_config --libmysqld-libs, --embedded-libs, 和 --embedded选项。**
  + **CMake的WITH\_EMBEDDED\_SERVER、WITH\_EMBEDDED\_SHARED\_LIBRARY和INSTALL\_SECURE\_FILE\_PRIV\_EMBEDDEDDIR选项。**
  + **mysql --server-arg选项（未记录）。**
  + **mysqltest --embedded-server, --server-arg, and --server-file选项**
  + **mysqltest\_embedded和mysql\_client\_test\_embedded测试程序**
* **mysql\_plugin工具被移除。替代品包括在服务器启动时使用--plugin-load或--plugin-load-add选项加载插件，或者在运行时使用INSTALL PLUGIN语句。**
* 移除 resolveip 工具。可以使用 nslookup、host 或 dig 来代替**。**
* resolve\_stack\_dump工具已被删除。来自官方MySQL构建的堆栈跟踪总是符号化的，因此没有必要使用 resolve\_stack\_dump**。**
* 以下服务器错误代码不被使用，并已被删除。专门测试任何这些错误的应用程序应该被更新。

ER\_BINLOG\_READ\_EVENT\_CHECKSUM\_FAILURE

ER\_BINLOG\_ROW\_RBR\_TO\_SBR

ER\_BINLOG\_ROW\_WRONG\_TABLE\_DEF

ER\_CANT\_ACTIVATE\_LOG

ER\_CANT\_CHANGE\_GTID\_NEXT\_IN\_TRANSACTION

ER\_CANT\_CREATE\_FEDERATED\_TABLE

ER\_CANT\_CREATE\_SROUTINE

ER\_CANT\_DELETE\_FILE

ER\_CANT\_GET\_WD

ER\_CANT\_SET\_GTID\_PURGED\_WHEN\_GTID\_MODE\_IS\_OFF

ER\_CANT\_SET\_WD

ER\_CANT\_WRITE\_LOCK\_LOG\_TABLE

ER\_CREATE\_DB\_WITH\_READ\_LOCK

ER\_CYCLIC\_REFERENCE

ER\_DB\_DROP\_DELETE

ER\_DELAYED\_NOT\_SUPPORTED

ER\_DIFF\_GROUPS\_PROC

ER\_DISK\_FULL

ER\_DROP\_DB\_WITH\_READ\_LOCK

ER\_DROP\_USER

ER\_DUMP\_NOT\_IMPLEMENTED

ER\_ERROR\_DURING\_CHECKPOINT

ER\_ERROR\_ON\_CLOSE

ER\_EVENTS\_DB\_ERROR

ER\_EVENT\_CANNOT\_DELETE

ER\_EVENT\_CANT\_ALTER

ER\_EVENT\_COMPILE\_ERROR

ER\_EVENT\_DATA\_TOO\_LONG

ER\_EVENT\_DROP\_FAILED

ER\_EVENT\_MODIFY\_QUEUE\_ERROR

ER\_EVENT\_NEITHER\_M\_EXPR\_NOR\_M\_AT

ER\_EVENT\_OPEN\_TABLE\_FAILED

ER\_EVENT\_STORE\_FAILED

ER\_EXEC\_STMT\_WITH\_OPEN\_CURSOR

ER\_FAILED\_ROUTINE\_BREAK\_BINLOG

ER\_FLUSH\_MASTER\_BINLOG\_CLOSED

ER\_FORM\_NOT\_FOUND

ER\_FOUND\_GTID\_EVENT\_WHEN\_GTID\_MODE\_IS\_OFF\_\_UNUSED

ER\_FRM\_UNKNOWN\_TYPE

ER\_GOT\_SIGNAL

ER\_GRANT\_PLUGIN\_USER\_EXISTS

ER\_GTID\_MODE\_REQUIRES\_BINLOG

ER\_GTID\_NEXT\_IS\_NOT\_IN\_GTID\_NEXT\_LIST

ER\_HASHCHK

ER\_INDEX\_REBUILD

ER\_INNODB\_NO\_FT\_USES\_PARSER

ER\_LIST\_OF\_FIELDS\_ONLY\_IN\_HASH\_ERROR

ER\_LOAD\_DATA\_INVALID\_COLUMN\_UNUSED

ER\_LOGGING\_PROHIBIT\_CHANGING\_OF

ER\_MALFORMED\_DEFINER

ER\_MASTER\_KEY\_ROTATION\_ERROR\_BY\_SE

ER\_NDB\_CANT\_SWITCH\_BINLOG\_FORMAT

ER\_NEVER\_USED

ER\_NISAMCHK

ER\_NO\_CONST\_EXPR\_IN\_RANGE\_OR\_LIST\_ERROR

ER\_NO\_FILE\_MAPPING

ER\_NO\_GROUP\_FOR\_PROC

ER\_NO\_RAID\_COMPILED

ER\_NO\_SUCH\_KEY\_VALUE

ER\_NO\_SUCH\_PARTITION\_\_UNUSED

ER\_OBSOLETE\_CANNOT\_LOAD\_FROM\_TABLE

ER\_OBSOLETE\_COL\_COUNT\_DOESNT\_MATCH\_CORRUPTED

ER\_ORDER\_WITH\_PROC

ER\_PARTITION\_SUBPARTITION\_ERROR

ER\_PARTITION\_SUBPART\_MIX\_ERROR

ER\_PART\_STATE\_ERROR

ER\_PASSWD\_LENGTH

ER\_QUERY\_ON\_MASTER

ER\_RBR\_NOT\_AVAILABLE

ER\_SKIPPING\_LOGGED\_TRANSACTION

ER\_SLAVE\_CHANNEL\_DELETE

ER\_SLAVE\_MULTIPLE\_CHANNELS\_HOST\_PORT

ER\_SLAVE\_MUST\_STOP

ER\_SLAVE\_WAS\_NOT\_RUNNING

ER\_SLAVE\_WAS\_RUNNING

ER\_SP\_GOTO\_IN\_HNDLR

ER\_SP\_PROC\_TABLE\_CORRUPT

ER\_SQL\_MODE\_NO\_EFFECT

ER\_SR\_INVALID\_CREATION\_CTX

ER\_TABLE\_NEEDS\_UPG\_PART

ER\_TOO\_MUCH\_AUTO\_TIMESTAMP\_COLS

ER\_UNEXPECTED\_EOF

ER\_UNION\_TABLES\_IN\_DIFFERENT\_DIR

ER\_UNSUPPORTED\_BY\_REPLICATION\_THREAD

ER\_UNUSED1

ER\_UNUSED2

ER\_UNUSED3

ER\_UNUSED4

ER\_UNUSED5

ER\_UNUSED6

ER\_VIEW\_SELECT\_DERIVED\_UNUSED

ER\_WRONG\_MAGIC

ER\_WSAS\_FAILED

* 删除了被废弃的 INFORMATION\_SCHEMA INNODB\_LOCKS 和 INNODB\_LOCK\_WAITS 表。使用Performance Schema data\_locks和data\_lock\_waits表代替。

注意

在MySQL 5.7中，INNODB\_LOCKS表中的LOCK\_TABLE列以及sys schema innodb\_lock\_waits和x$innodb\_lock\_waits视图中的locked\_table列包含组合模式/表名值。在MySQL 8.0中，data\_locks表和sys schema视图包含单独的模式名和表名列。参见第28.4.3.9节，"innodb\_lock\_waits和x$innodb\_lock\_waits视图"。

* InnoDB不再支持压缩的临时表。当启用innodb\_strict\_mode时（默认），如果指定了ROW\_FORMAT=COMPRESSED或KEY\_BLOCK\_SIZE，CREATE TEMPORARY TABLE会返回一个错误。如果innodb\_strict\_mode被禁用，会发出警告，并且使用非压缩行格式创建临时表。
* 在MySQL数据目录之外创建表空间数据文件时，InnoDB不再创建.isl文件（InnoDB符号链接文件）。innodb\_directories选项现在支持定位在数据目录之外创建的表空间文件。

随着这一变化，不再支持在服务器离线时通过手动修改.isl文件移动远程表空间。移动远程表空间文件现在被innodb\_directories选项所支持。参见第15.6.3.6节，"在服务器离线时移动表空间文件"。

* 删除了下列 InnoDB 文件格式变量。
  + innodb\_file\_format
  + innodb\_file\_format\_check
  + innodb\_file\_format\_max
  + innodb\_large\_prefix

文件格式变量对于创建与MySQL 5.1中InnoDB早期版本兼容的表是必要的。现在，MySQL 5.1已经达到其产品生命周期的终点，这些选项不再需要了。

FILE\_FORMAT列已从INNODB\_TABLES和INNODB\_TABLESPACES信息模式表中删除。

* 移除了innodb\_support\_xa系统变量，该变量能够支持XA事务中的两阶段提交。InnoDB对XA事务中的两阶段提交的支持总是被启用。
* 删除了对 DTrace 的支持。
* 删除了JSON\_APPEND()函数。使用JSON\_ARRAY\_APPEND()代替。
* 在MySQL 8.0.13中删除了对在共享InnoDB表空间中放置表分区的支持。共享表空间包括InnoDB系统表空间和一般表空间。关于识别共享表空间中的分区并将其移至按表文件的表空间的信息，请参见第2.11.5节 "为升级准备你的安装"。
* 在MySQL 8.0.13中，对在SET以外的语句中设置用户变量的支持已被废弃。这一功能将在MySQL 9.0中删除。
* --ndb perror选项已被删除。请使用ndb\_perror工具来代替。
* 移除innodb\_undo\_logs变量。innodb\_rollback\_segments变量执行相同的功能，应该使用它来代替。
* Innodb\_available\_undo\_logs状态变量被移除。每个表空间的可用回滚段的数量可以使用SHOW VARIABLES LIKE 'innodb\_rollback\_segments'来检索。
* 从MySQL 8.0.14开始，以前废弃的innodb\_undo\_tablespaces变量不再可配置。更多信息，见第15.6.3.4节 "undo tablespaces"。
* 支持ALTER TABLE ... UPGRADE PARTITIONING语句的支持已被删除。
* 从MySQL 8.0.16开始，对internal\_tmp\_disk\_storage\_engine系统变量的支持已被删除；磁盘上的内部临时表现在总是使用InnoDB存储引擎。更多信息请参见磁盘上内部临时表的存储引擎。
* DISABLE\_SHARED CMake 选项未被使用，现已被删除

## **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**

本节列出了服务器变量、状态变量以及在MySQL 8.0中首次添加、被废弃或被删除的选项。

**MySQL 8.0中引入的选项和变量**

下列系统变量、状态变量和服务器选项已在MySQL 8.0中添加。

* Acl\_cache\_items\_count。缓存的特权对象的数量。在MySQL 8.0.0中添加。
* Audit\_log\_current\_size。审计日志文件的当前大小。在MySQL 8.0.11中添加。
* Audit\_log\_event\_max\_drop\_size。最大丢弃的审计事件的大小。在MySQL 8.0.11中添加。
* Audit\_log\_events。处理的审计事件的数量。在MySQL 8.0.11中添加。
* Audit\_log\_events\_filtered。被过滤的审计事件的数量。在MySQL 8.0.11中添加。
* Audit\_log\_events\_lost。被丢弃的审计事件的数量。在MySQL 8.0.11中添加。
* Audit\_log\_events\_written。写入的审计事件的数量。在MySQL 8.0.11中添加。
* Audit\_log\_total\_size。写入的审计事件的综合大小。在MySQL 8.0.11中添加。
* Audit\_log\_write\_waits。写入延迟的审计事件的数量。在MySQL 8.0.11中添加。
* Authentication\_ldap\_sasl\_supported\_methods。为SASL LDAP认证支持的认证方法。在MySQL 8.0.21中添加。
* Caching\_sha2\_password\_rsa\_public\_key：缓存\_sha2\_password认证插件RSA公钥值。在MySQL 8.0.4中添加。
* Com\_alter\_resource\_group。ALTER RESOURCE GROUP语句的计数。在MySQL 8.0.3中添加。
* [**Com\_alter\_user\_default\_role**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx):：ALTER USER ..: ALTER USER ... DEFAULT ROLE语句的计数。在MySQL 8.0.0中添加
* Com\_clone。CLONE语句的数量。在MySQL 8.0.2中添加
* Com\_create\_resource\_group: CREATE RESOURCE GROUP语句的计数。在MySQL 8.0.3中添加。
* om\_create\_role: CREATE ROLE语句的计数。在MySQL 8.0.0中添加。
* Com\_drop\_resource\_group: DROP RESOURCE GROUP语句的计数。在MySQL 8.0.3中添加。
* Com\_drop\_role: DROP ROLE语句的计数。在MySQL 8.0.0中添加。
* Com\_grant\_roles: GRANT ROLE语句的计数。在MySQL 8.0.0中添加。
* Com\_install\_component: INSTALL COMPONENT语句的计数。在MySQL 8.0.0中添加。
* Com\_replica\_start。START REPLICA和START SLAVE语句的计数。在MySQL 8.0.22中添加。
* Com\_replica\_stop。STOP REPLICA和STOP SLAVE语句的数量。在MySQL 8.0.22中添加。
* Com\_restart。RESTART语句的计数。在MySQL 8.0.4中添加。
* Com\_revoke\_roles。REVOKE ROLES语句的计数。在MySQL 8.0.0中添加。
* Com\_set\_resource\_group: SET RESOURCE GROUP语句的数量。在MySQL 8.0.3中添加。
* Com\_set\_role: SET ROLE语句的计数。在MySQL 8.0.0中添加。
* [**Com\_show\_replica\_status**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx): 。SHOW REPLICA STATUS和SHOW SLAVE STATUS语句的计数。在MySQL 8.0.22中添加。
* Com\_show\_replicas。SHOW REPLICAS和SHOW SLAVE HOSTS语句的计数。在MySQL 8.0.22中添加。
* Com\_uninstall\_component。UINSTALL COMPONENT语句的计数。在MySQL 8.0.0中添加。
* Compression\_algorithm。当前连接的压缩算法。在MySQL 8.0.18中添加。
* Compression\_level（压缩级别）。当前连接的压缩级别。在MySQL 8.0.18中添加。
* Connection\_control\_delay\_generated。服务器延迟了多少次连接请求。在MySQL 8.0.1中添加。
* Current\_tls\_ca。ssl\_ca系统变量的当前值。在MySQL 8.0.16中添加。
* Current\_tls\_capath。ssl\_capath系统变量的当前值。在MySQL 8.0.16中添加。
* [**Current\_tls\_cert**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Current_tls_cert):。ssl\_cert系统变量的当前值。在MySQL 8.0.16中添加。
* [**Current\_tls\_cipher**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Current_tls_cipher):。ssl\_cipher系统变量的当前值。在MySQL 8.0.16中添加。
* Current\_tls\_ciphersuites。tsl\_ciphersuites系统变量的当前值。在MySQL 8.0.16中添加。
* Current\_tls\_crl。ssl\_crl系统变量的当前值。在MySQL 8.0.16中添加。
* Current\_tls\_crlpath。ssl\_crlpath系统变量的当前值。在MySQL 8.0.16中添加。
* Current\_tls\_key。ssl\_key系统变量的当前值。在MySQL 8.0.16中添加。
* [**Current\_tls\_version**](file:///C:\Users\Administrator\Downloads\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Current_tls_version):。tls\_version系统变量的当前值。在MySQL 8.0.16中添加。
* Error\_log\_buffered\_bytes。在error\_log表中使用的字节数。在MySQL 8.0.22中添加。
* Error\_log\_buffered\_events。在error\_log表中的事件数。在MySQL 8.0.22中添加。
* Error\_log\_expired\_events。从error\_log表中丢弃的事件的数量。在MySQL 8.0.22中添加。
* Error\_log\_latest\_write。最后一次写到error\_log表的时间。在MySQL 8.0.22中添加。
* Firewall\_access\_denied。被MySQL企业防火墙拒绝的语句数量。在MySQL 8.0.11中添加。
* Firewall\_access\_granted。被MySQL企业防火墙接受的语句数量。在MySQL 8.0.11中添加。
* Firewall\_cached\_entries。由MySQL企业防火墙记录的语句数量。在MySQL 8.0.11中添加。
* Innodb\_redo\_log\_enabled。InnoDBredo log状态。在MySQL 8.0.21中添加。
* Innodb\_system\_rows\_deleted。从系统模式表中删除的行数。在MySQL 8.0.19中添加。
* Innodb\_system\_rows\_inserted。插入到系统模式表中的行数。在MySQL 8.0.19中添加。
* Innodb\_system\_rows\_read。从系统模式表中读取的行数。在MySQL 8.0.19中添加。Innodb\_undo\_tablespaces\_active。活动的undo tablespaces的数量。在MySQL 8.0.14中添加。
* Innodb\_undo\_tablespaces\_explicit。用户创建的undo tablespaces的数量。在MySQL 8.0.14中添加。
* Innodb\_undo\_tablespaces\_implicit。由InnoDB创建的undo tablespaces的数量。在MySQL 8.0.14中添加
* Innodb\_undo\_tablespaces\_total：撤消的表空间总数。在MySQL 8.0.14中添加。
* Mysqlx\_bytes\_received\_compressed\_payload。收到的作为压缩消息有效载荷的字节数，在解压前测量。在MySQL 8.0.19中添加。
* Mysqlx\_bytes\_received\_uncompressed\_frame: 作为压缩消息有效载荷收到的字节数，在解压后测量。在MySQL 8.0.19中添加。
* Mysqlx\_bytes\_sent\_compressed\_payload。作为压缩的消息有效载荷发送的字节数，在压缩后测量。在MySQL 8.0.19中添加。
* Mysqlx\_bytes\_sent\_uncompressed\_frame: 作为压缩消息有效载荷发送的字节数，在压缩前测量。在MySQL 8.0.19中添加
* Mysqlx\_compression\_algorithm。此会话的X协议连接使用的压缩算法。在MySQL 8.0.20中添加。
* Mysqlx\_compression\_level: 此会话的X协议连接使用的压缩级别。在MySQL 8.0.20中添加。
* Secondary\_engine\_execution\_count。卸载到二级引擎的查询数量。在MySQL 8.0.13中添加。
* activate\_all\_roles\_on\_login: 是否在连接时激活所有用户角色。在MySQL 8.0.2中添加
* admin-ssl：启用连接加密。在MySQL 8.0.21中添加。
* admin\_address。在管理界面上连接时要绑定的IP地址。在MySQL 8.0.14中添加。
* admin\_port。用于管理界面连接的TCP/IP号码。在MySQL 8.0.14中添加。
* admin\_ssl\_ca。包含受信任的SSL证书颁发机构列表的文件。在MySQL 8.0.21中添加
* admin\_ssl\_capath。包含可信的SSL证书颁发机构证书文件的目录。在MySQL 8.0.21中添加。
* admin\_ssl\_cert。包含X.509证书的文件。在MySQL 8.0.21中添加
* admin\_ssl\_cipher。允许用于连接加密的密码。在MySQL 8.0.21中添加。
* admin\_ssl\_crl。包含证书撤销列表的文件。在MySQL 8.0.21中添加。
* admin\_ssl\_crlpath。包含证书撤销列表文件的目录。在MySQL 8.0.21中添加。
* admin\_ssl\_key。包含X.509密钥的文件。在MySQL 8.0.21中添加
* admin\_tls\_ciphersuites。允许的用于加密连接的TLSv1.3密码套件。在MySQL 8.0.21中添加。
* admin\_tls\_version。允许用于加密连接的TLS协议。在MySQL 8.0.21中添加。
* audit-log。是否激活审计日志插件。在MySQL 8.0.11中添加。
* audit\_log\_buffer\_size。审计日志缓冲区的大小。在MySQL 8.0.11中添加。
* audit\_log\_compression。审计日志文件的压缩方法。在MySQL 8.0.11中添加。
* audit\_log\_connection\_policy。连接相关事件的审计日志策略。在MySQL 8.0.11中添加。
* audit\_log\_current\_session。是否审计当前会话。在MySQL 8.0.11中添加。
* audit\_log\_encryption。审计日志文件的加密方法。在MySQL 8.0.11中添加。
* audit\_log\_exclude\_accounts。不予审计的账户。在MySQL 8.0.11中添加。
* audit\_log\_file。审计日志文件的名称。在MySQL 8.0.11中添加。
* audit\_log\_filter\_id。当前审计日志过滤器的ID。在MySQL 8.0.11中添加。
* audit\_log\_flush。关闭并重新打开审计日志文件。在MySQL 8.0.11中添加。
* audit\_log\_format。审计日志文件格式。在MySQL 8.0.11中添加。
* audit\_log\_include\_accounts。要审计的账户。在MySQL 8.0.11中添加
* audit\_log\_max\_size。对JSON审计日志文件的组合大小的限制。在MySQL 8.0.25中添加。
* audit\_log\_password\_history\_keep\_days。保留存档的审计日志加密密码的天数。在MySQL 8.0.17中添加。
* audit\_log\_policy。审计日志策略。在MySQL 8.0.11中添加。
* audit\_log\_prune\_seconds。审计日志文件被修剪的秒数。在MySQL 8.0.24中添加。
* audit\_log\_read\_buffer\_size。审计日志文件的读取缓冲区大小。在MySQL 8.0.11中添加。
* audit\_log\_rotate\_on\_size。关闭并重新打开这个大小的审计日志文件。在MySQL 8.0.11中添加。
* audit\_log\_statement\_policy。语句相关事件的审计日志策略。在MySQL 8.0.11中添加。
* audit\_log\_strategy。审计日志策略。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_auth\_method\_name：认证方法名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_bind\_base\_dn。LDAP服务器的基本区分名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_bind\_root\_dn。LDAP服务器根部的识别名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_bind\_root\_pwd。LDAP服务器根绑定密码。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_ca\_path。LDAP服务器证书授权文件名。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_group\_search\_attr。LDAP服务器组搜索属性。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_group\_search\_filter。LDAP自定义组搜索过滤器。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_init\_pool\_size。LDAP服务器初始连接池大小。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_log\_status。LDAP服务器的日志级别。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_max\_pool\_size。LDAP服务器的最大连接池大小。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_referral。是否启用LDAP搜索转介。在MySQL 8.0.20中添加。
* authentication\_ldap\_sasl\_server\_host：LDAP服务器主机名或IP地址。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_server\_port。LDAP服务器的端口号。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_tls。是否使用与LDAP服务器的加密连接。在MySQL 8.0.11中添加。
* authentication\_ldap\_sasl\_user\_search\_attr。LDAP服务器的用户搜索属性。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_auth\_method\_name：认证方法名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_bind\_base\_dn。LDAP服务器的基本区分名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_bind\_root\_dn。LDAP服务器根部的识别名称。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_bind\_root\_pwd。LDAP服务器根绑定密码。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_ca\_path。LDAP服务器证书授权文件名。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_group\_search\_attr。LDAP服务器组搜索属性。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_group\_search\_filter。LDAP自定义组搜索过滤器。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_init\_pool\_size。LDAP服务器初始连接池大小。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_log\_status。LDAP服务器的日志级别。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_max\_pool\_size。LDAP服务器的最大连接池大小。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_referral。是否启用LDAP搜索转介。在MySQL 8.0.20中添加。
* authentication\_ldap\_simple\_server\_host：LDAP服务器主机名或IP地址。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_server\_port。LDAP服务器的端口号。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_tls。是否使用与LDAP服务器的加密连接。在MySQL 8.0.11中添加。
* authentication\_ldap\_simple\_user\_search\_attr。LDAP服务器的用户搜索属性。在MySQL 8.0.11中添加。
* authentication\_windows\_log\_level。Windows认证插件的日志级别。在MySQL 8.0.11中添加。
* authentication\_windows\_use\_principal\_name：是否使用Windows认证插件的主体名称。在MySQL 8.0.11中添加
* binlog\_encryption。启用对该服务器上的二进制日志文件和中继日志文件的加密。在MySQL 8.0.14中添加。
* binlog\_expire\_logs\_seconds。在此秒数后清除二进制日志。在MySQL 8.0.1中添加
* binlog\_rotate\_encryption\_master\_key\_at\_startup。在服务器启动时旋转二进制日志主密钥。在MySQL 8.0.14中添加。
* binlog\_row\_metadata。当使用基于行的日志时，是否将所有或仅最小的表相关元数据记录到二进制日志。在MySQL 8.0.1中添加。
* binlog\_row\_value\_options。启用基于行的复制的部分JSON更新的二进制日志。在MySQL 8.0.3中添加。
* binlog\_transaction\_compression。启用二进制日志文件中事务有效载荷的压缩。在MySQL 8.0.20中添加。
* binlog\_transaction\_compression\_level\_zstd：二进制日志文件中事务有效载荷的压缩级别。在MySQL 8.0.20中添加。
* binlog\_transaction\_dependency\_history\_size。为查找最后更新某行的事务而保留的行哈希值的数量。在MySQL 8.0.1中添加。
* binlog\_transaction\_dependency\_tracking。依赖性信息的来源（提交时间戳或事务写入集），据此评估哪些事务可以由副本的多线程应用者并行执行。在MySQL 8.0.1中添加。
* caching\_sha2\_password\_auto\_generate\_rsa\_keys。是否自动生成RSA密钥对文件。在MySQL 8.0.4中添加。
* caching\_sha2\_password\_digest\_rounds。caching\_sha2\_password认证插件的哈希轮数。在MySQL 8.0.24中添
* caching\_sha2\_password\_private\_key\_path。SHA2认证插件的私钥路径名称。在MySQL 8.0.3中添加。
* caching\_sha2\_password\_public\_key\_path。SHA2认证插件的公钥路径名称。在MySQL 8.0.3中添加。
* clone\_autotune\_concurrency。为远程克隆操作启用动态生成线程。在MySQL 8.0.17中添加
* clone\_buffer\_size。定义捐赠者MySQL服务器实例上的中间缓冲区的大小。在MySQL 8.0.17中添加。
* clone\_ddl\_timeout。克隆操作等待备份锁的秒数。在MySQL 8.0.17中添加。
* clone\_donor\_timeout\_after\_network\_failure。网络故障后重新启动克隆操作的允许时间。在MySQL 8.0.24中添加。
* clone\_enable\_compression。在克隆过程中启用网络层的数据压缩。在MySQL 8.0.17中添加。
* clone\_max\_concurrency。用于执行克隆操作的最大并发线程数。在MySQL 8.0.17中添加。
* clone\_max\_data\_bandwidth：用于远程克隆操作的最大数据传输速率，单位为MiB/s。在MySQL 8.0.17中添加。
* clone\_max\_network\_bandwidth：远程克隆操作的最大网络传输速率，单位为每秒MiB。在MySQL 8.0.17中添加。
* clone\_ssl\_ca。指定证书授权（CA）文件的路径。在MySQL 8.0.14中添加。
* clone\_ssl\_cert: 指定公钥证书文件的路径。在MySQL 8.0.14中添加。
* clone\_ssl\_key。指定私钥文件的路径。在MySQL 8.0.14中添加。
* clone\_valid\_donor\_list。定义远程克隆操作的捐赠者主机地址。在MySQL 8.0.17中添加。
* connection\_control\_failed\_connections\_threshold。延迟发生前连续失败的连接尝试。在MySQL 8.0.1中添加。
* connection\_control\_max\_connection\_delay。服务器响应失败连接尝试的最大延迟（毫秒）。在MySQL 8.0.1中添加。
* connection\_control\_min\_connection\_delay。服务器响应失败的连接尝试的最小延迟（毫秒）。在MySQL 8.0.1中添加。
* create\_admin\_listener\_thread。是否为管理界面的连接使用专用监听线程。在MySQL 8.0.14中添加。
* cte\_max\_recursion\_depth。公共表表达式的最大递归深度。在MySQL 8.0.3中添加。
* ddl-rewriter。是否激活ddl\_rewriter插件。在MySQL 8.0.16中添加。
* default\_collation\_for\_utf8mb4：utf8mb4字符集的默认排序。在MySQL 8.0.11中添加。
* default\_table\_encryption。默认的模式和表空间加密设置。在MySQL 8.0.16中添加。
* dragnet.Status。对dragnet.log\_error\_filter\_rules的最近一次分配的结果。在MySQL 8.0.12中添加。
* dragnet.log\_error\_filter\_rules。用于错误日志的过滤规则。在MySQL 8.0.4中添加。
* early-plugin-load。指定在加载强制性内置插件和存储引擎初始化前加载的插件。在MySQL 8.0.0中添加。
* generated\_random\_password\_length。生成密码的最大长度。在MySQL 8.0.18中添加。
* group\_replication\_advertise\_recovery\_endpoints。为分布式恢复提供的连接。在MySQL 8.0.21中添加。
* group\_replication\_autorejoin\_tries。成员为自动重新加入组而进行的尝试次数。在MySQL 8.0.16中添加。
* group\_replication\_clone\_threshold。捐赠者和接受者之间的事务数差距，超过这个差距就会使用远程克隆操作进行状态转移。在MySQL 8.0.17中添加。
* group\_replication\_communication\_debug\_options。组复制组件的调试信息的级别。在MySQL 8.0.3中添加。
* group\_replication\_communication\_max\_message\_size。组复制通信的最大消息大小，较大的消息会被分割。在MySQL 8.0.16中添加。
* group\_replication\_consistency。组提供的事务一致性保证的类型。在MySQL 8.0.14中添加。
* group\_replication\_exit\_state\_action。实例非自愿离开组时的行为方式。在MySQL 8.0.12中添加。
* group\_replication\_flow\_control\_hold\_percent。保持未使用的组配额的百分比。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_max\_commit\_quota。组的最大流量控制配额。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_member\_quota\_percent。在计算配额时，成员应假定自己可用的配额的百分比。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_min\_quota。可分配给每个成员的最低流量控制配额。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_min\_recovery\_quota。可分配给每个成员的最低配额，因为另一个组成员正在恢复。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_period。定义流量控制迭代之间要等待多少秒。在MySQL 8.0.2中添加。
* group\_replication\_flow\_control\_release\_percent。当流量控制不再需要节制写作者成员时，应如何释放组配额。在MySQL 8.0.2中添加。
* group\_replication\_ip\_allowlist。允许连接到组的主机列表（MySQL 8.0.22及以后版本）。在MySQL 8.0.22中添加。
* group\_replication\_member\_expel\_timeout。怀疑群组成员失败和将其从群组驱逐之间的时间，导致群组成员重新配置。在MySQL 8.0.13中添加。
* group\_replication\_member\_weight。这个成员被选为主要成员的几率。在MySQL 8.0.2中添加。
* group\_replication\_message\_cache\_size。组通信引擎消息缓存的最大内存（XCom）。在MySQL 8.0.16中添加。
* group\_replication\_recovery\_compression\_algorithms。允许的出站恢复连接的压缩算法。在MySQL 8.0.18中添加。
* group\_replication\_recovery\_get\_public\_key。是否接受关于从捐赠者获取公钥的首选项。在MySQL 8.0.4中添加。
* group\_replication\_recovery\_public\_key\_path。用于接受公钥信息。在MySQL 8.0.4中添加。
* group\_replication\_recovery\_tls\_ciphersuites。当TLSv1.3被用于与该实例作为客户端（加入成员）的连接加密时，允许的密码套件。在MySQL 8.0.19中添加。
* group\_replication\_recovery\_tls\_version。允许的TLS协议用于作为客户端（加入成员）的连接加密。在MySQL 8.0.19中添加。
* group\_replication\_recovery\_zstd\_compression\_level。使用zstd压缩的恢复连接的压缩级别。在MySQL 8.0.18中添加。
* group\_replication\_tls\_source。组复制的TLS材料的来源。在MySQL 8.0.21中添加。
* group\_replication\_unreachable\_majority\_timeout。导致少数人离开组的网络分区要等待多长时间。在MySQL 8.0.2中添加。
* histogram\_generation\_max\_mem\_size。用于创建直方图统计的最大内存。在MySQL 8.0.2中添加。
* immediate\_server\_version。作为即时复制源的服务器的MySQL服务器版本号。在MySQL 8.0.14中添加。
* information\_schema\_stats\_expiry。缓存表统计的过期设置。在MySQL 8.0.3中添加。
* innodb\_buffer\_pool\_debug。当缓冲池的大小小于1GB时，允许多个缓冲池实例。在MySQL 8.0.0中添加。
* innodb\_buffer\_pool\_in\_core\_file。控制将缓冲池页面写入核心文件。在MySQL 8.0.14中添加。
* innodb\_checkpoint\_disabled。禁用检查点，以便故意的服务器退出总是启动恢复。在MySQL 8.0.2中添加。
* innodb\_ddl\_log\_crash\_reset\_debug。调试选项，用于重置DDL日志崩溃注入计数器。在MySQL 8.0.3中添加。
* innodb\_deadlock\_detect。启用或禁用死锁检测。在MySQL 8.0.0中添加。
* innodb\_dedicated\_server。启用缓冲池大小、日志文件大小和冲刷方法的自动配置。在MySQL 8.0.3中添加。
* innodb\_directories。定义了在启动时扫描表空间数据文件的目录。在MySQL 8.0.4中添加。
* innodb\_doublewrite\_batch\_size。每批写入的双写页的数量。在MySQL 8.0.20中添加。
* innodb\_doublewrite\_dir。双重写缓冲区文件目录。在MySQL 8.0.20中添加。
* innodb\_doublewrite\_files。双重写文件的数量。在MySQL 8.0.20中添加。
* innodb\_doublewrite\_pages。每个线程的双写页数。在MySQL 8.0.20中添加。
* innodb\_extend\_and\_initialize。控制在Linux上如何分配新的表空间页。在MySQL 8.0.22中添加。
* innodb\_fsync\_threshold。控制InnoDB在创建新文件时调用fsync的频率。在MySQL 8.0.13中添加。
* innodb\_idle\_flush\_pct。限制InnoDB空闲时的I/0操作。在MySQL 8.0.18中添加。
* innodb\_log\_checkpoint\_fuzzy\_now。调试选项，强制InnoDB写入模糊检查点。在MySQL 8.0.13中添加。
* innodb\_log\_spin\_cpu\_abs\_lwm。CPU使用量的最小值，低于此值时，用户线程在等待刷新的重做时不再旋转。在MySQL 8.0.11中添加。
* innodb\_log\_spin\_cpu\_pct\_hwm。用户线程在等待刷新的重做时不再旋转的最大CPU使用率。在MySQL 8.0.11中添加。
* innodb\_log\_wait\_for\_flush\_spin\_hwm。最大的平均日志刷新时间，超过该时间用户线程在等待刷新的重做时不再旋转。在MySQL 8.0.11中添加。
* innodb\_log\_writer\_threads。启用专用的日志写入器线程来写入和冲刷redo log。在MySQL 8.0.22中添加。
* innodb\_parallel\_read\_threads。用于并行索引读取的线程数量。在MySQL 8.0.14中添加。
* innodb\_print\_ddl\_logs。是否将DDL日志打印到错误日志。在MySQL 8.0.3中添加。
* innodb\_redo\_log\_archive\_dirs。标记的redo log存档目录。在MySQL 8.0.17中添加。
* innodb\_redo\_log\_encrypt。控制对加密表空间的redo log数据进行加密。在MySQL 8.0.1中添加。
* innodb\_scan\_directories。定义在InnoDB恢复期间扫描表空间文件的目录。在MySQL 8.0.2中添加。
* innodb\_segment\_reserve\_factor。保留为空页的表空间文件段页的百分比。在MySQL 8.0.25中添加。
* innodb\_spin\_wait\_pause\_multiplier。用于确定自旋等待循环中PAUSE指令数量的乘数值。在MySQL 8.0.16中添加。
* innodb\_stats\_include\_delete\_marked。在计算持久性InnoDB统计数据时，包括有删除标记的记录。在MySQL 8.0.1中添加。
* innodb\_temp\_tablespaces\_dir。会话临时表空间路径。在MySQL 8.0.13中添加。
* innodb\_tmpdir。在线ALTER TABLE操作期间创建的临时表文件的目录位置。在MySQL 8.0.0中添加。
* innodb\_undo\_log\_encrypt。控制对加密表空间的Undo logs数据的加密。在MySQL 8.0.1中添加。
* innodb\_use\_fdatasync。当向操作系统冲刷数据时，InnoDB是否使用fdatasync()而不是fsync()。在MySQL 8.0.25中添加。
* innodb\_validate\_tablespace\_paths。在启动时启用表空间路径验证。在MySQL 8.0.21中添加。
* internal\_tmp\_mem\_storage\_engine：用于内部内存临时表的存储引擎。在MySQL 8.0.2中添加。
* keyring-migration-destination: 钥匙迁移目的地钥匙圈插件。在MySQL 8.0.4中添加。
* keyring-migration-host：连接到运行中的服务器进行密钥迁移的主机名。在MySQL 8.0.4中添加。
* keyring-migration-password：连接到运行中的服务器进行密钥迁移的密码。在MySQL 8.0.4中添加。
* keyring-migration-port。TCP/IP端口号，用于连接到运行中的服务器进行密钥迁移。在MySQL 8.0.4中添加。
* keyring-migration-socket。Unix套接字文件或Windows命名的管道，用于连接到运行中的服务器进行密钥迁移。在MySQL 8.0.4中添加。
* keyring-migration-source。钥匙迁移源钥匙圈插件。在MySQL 8.0.4中添加。
* keyring-migration-to-component。钥匙圈迁移是指从插件到组件。在MySQL 8.0.24中添加。
* keyring-migration-user: 用于连接到运行中的服务器进行密钥迁移的用户名。在MySQL 8.0.4中添加。
* keyring\_aws\_cmk\_id。AWS钥匙圈插件客户主钥匙的ID值。在MySQL 8.0.11中添加。
* keyring\_aws\_conf\_file。AWS钥匙圈插件的配置文件位置。在MySQL 8.0.11中添加。
* keyring\_aws\_data\_file。AWS钥匙圈插件的存储文件位置。在MySQL 8.0.11中添加。
* keyring\_aws\_region。AWS钥匙圈插件区域。在MySQL 8.0.11中添加。
* keyring\_encrypted\_file\_data：keyring\_encrypted\_file插件数据文件。在MySQL 8.0.11中添加。
* keyring\_encrypted\_file\_password：keyring\_encrypted\_file插件的密码。在MySQL 8.0.11中添加。
* keyring\_hashicorp\_auth\_path。HashiCorp Vault AppRole认证路径。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_ca\_path。keyring\_hashicorp CA文件的路径。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_caching：是否启用keyring\_hashicorp缓存。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_auth\_path：使用中的keyring\_hashicorp\_auth\_path值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_ca\_path：使用中的keyring\_hashicorp\_ca\_path值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_caching：使用中的keyring\_hashicorp\_caching值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_role\_id：使用中的keyring\_hashicorp\_role\_id值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_server\_url：使用中的keyring\_hashicorp\_server\_url值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_commit\_store\_path：使用中的keyring\_hashicorp\_store\_path值。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_role\_id。HashiCorp Vault AppRole认证角色ID。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_secret\_id。HashiCorp Vault AppRole认证秘密ID。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_server\_url。HashiCorp Vault服务器的URL。在MySQL 8.0.18中添加。
* keyring\_hashicorp\_store\_path。HashiCorp Vault的存储路径。在MySQL 8.0.18中添加。
* keyring\_oci\_ca\_certificate。用于同行认证的CA证书文件。在MySQL 8.0.22中添加。
* keyring\_oci\_compartment。OCI隔间OCID。在MySQL 8.0.22中添加。
* keyring\_oci\_encryption\_endpoint。OCI加密服务器端点。在MySQL 8.0.22中添加。
* keyring\_oci\_key\_file。OCI RSA私钥文件。在MySQL 8.0.22中添加。
* keyring\_oci\_key\_fingerprint。OCI RSA私钥文件的指纹。在MySQL 8.0.22中添加。
* keyring\_oci\_management\_endpoint。OCI管理服务器端点。在MySQL 8.0.22中添加。
* keyring\_oci\_master\_key。OCI主密钥OCID。在MySQL 8.0.22中添加。
* keyring\_oci\_secrets\_endpoint。OCI机密服务器的端点。在MySQL 8.0.22中添加。
* keyring\_oci\_tenancy。OCI租户OCID。在MySQL 8.0.22中添加。
* keyring\_oci\_user。OCI用户的OCID。在MySQL 8.0.22中添加。
* keyring\_oci\_vaults\_endpoint。OCI金库服务器端点。在MySQL 8.0.22中添加。
* keyring\_oci\_virtual\_vault。OCI保险库OCID。在MySQL 8.0.22中添加。
* keyring\_okv\_conf\_dir。Oracle Key Vault密钥库插件配置目录。在MySQL 8.0.11中添加。
* keyring\_operations。是否启用了钥匙圈操作。在MySQL 8.0.4中添加。
* lock\_order。是否在运行时启用LOCK\_ORDER工具。在MySQL 8.0.17中添加。
* lock\_order\_debug\_loop。当LOCK\_ORDER工具遇到标记为循环的依赖关系时，是否引起调试断言。在MySQL 8.0.17中添加。
* lock\_order\_debug\_missing\_arc: 当LOCK\_ORDER工具遇到未声明的依赖关系时，是否会引起调试断言。在MySQL 8.0.17中添加。
* lock\_order\_debug\_missing\_key。当LOCK\_ORDER工具遇到没有正确使用Performance Schema工具的对象时，是否会引起调试断言。在MySQL 8.0.17中添加。
* lock\_order\_debug\_missing\_unlock。当LOCK\_ORDER工具遇到仍然持有的锁被破坏时，是否会引起调试断言。在MySQL 8.0.17中添加。
* lock\_order\_dependencies。lock\_order\_dependencies.txt文件的路径。在MySQL 8.0.17中添加。
* lock\_order\_extra\_dependencies。第二个依赖文件的路径。在MySQL 8.0.17中添加。
* lock\_order\_output\_directory。LOCK\_ORDER工具写入日志的目录。在MySQL 8.0.17中添加。
* lock\_order\_print\_txt。是否执行锁序图分析和打印文本报告。在MySQL 8.0.17中添加。
* lock\_order\_trace\_loop。当LOCK\_ORDER工具遇到标记为循环的依赖关系时，是否打印日志文件跟踪。在MySQL 8.0.17中添加。
* lock\_order\_trace\_missing\_arc：当LOCK\_ORDER工具遇到未声明的依赖关系时，是否打印日志文件跟踪。在MySQL 8.0.17中添加。
* lock\_order\_trace\_missing\_key。当LOCK\_ORDER工具遇到没有正确使用Performance Schema工具的对象时，是否打印日志文件跟踪。在MySQL 8.0.17中添加。
* lock\_order\_trace\_missing\_unlock。当LOCK\_ORDER工具遇到仍然持有的锁被破坏时，是否打印日志文件跟踪。在MySQL 8.0.17中添加。
* log\_error\_filter\_rules。用于错误日志的过滤规则。在MySQL 8.0.2中添加。
* log\_error\_services。用于错误日志记录的组件。在MySQL 8.0.2中添加。
* log\_error\_suppression\_list。要抑制的警告/信息错误日志信息。在MySQL 8.0.13中添加。
* log\_slow\_extra。是否向慢速查询日志文件写入额外信息。在MySQL 8.0.14中添加。
* mandatory\_roles。自动授予所有用户的角色。在MySQL 8.0.2中添加。
* mysql\_firewall\_mode。MySQL企业防火墙是否运行。在MySQL 8.0.11中添加。
* mysql\_firewall\_trace。是否启用防火墙跟踪。在MySQL 8.0.11中添加。
* mysqlx：是否初始化了X插件。在MySQL 8.0.11中添加。
* mysqlx\_compression\_algorithms。允许用于X协议连接的压缩算法。在MySQL 8.0.19中添加。
* mysqlx\_deflate\_default\_compression\_level。X协议连接上Deflate算法的默认压缩级别。在MySQL 8.0.20中添加。
* mysqlx\_deflate\_max\_client\_compression\_level。X协议连接上的Deflate算法的最大允许压缩级别。在MySQL 8.0.20中添加。
* mysqlx\_interactive\_timeout。等待交互式客户端超时的秒数。在MySQL 8.0.4中添加。
* mysqlx\_lz4\_default\_compression\_level。X协议连接上LZ4算法的默认压缩级别。在MySQL 8.0.20中添加。
* mysqlx\_lz4\_max\_client\_compression\_level。X协议连接上LZ4算法的最大允许压缩级别。在MySQL 8.0.20中添加。
* mysqlx\_read\_timeout。等待阻塞读操作完成的秒数。在MySQL 8.0.4中添加。
* mysqlx\_wait\_timeout。等待连接活动的秒数。在MySQL 8.0.4中添加。
* mysqlx\_write\_timeout。等待阻塞的写操作完成的秒数。在MySQL 8.0.4中添加。
* mysqlx\_zstd\_default\_compression\_level。X协议连接上zstd算法的默认压缩级别。在MySQL 8.0.20中添加。
* mysqlx\_zstd\_max\_client\_compression\_level。X协议连接上zstd算法的最大允许压缩级别。在MySQL 8.0.20中添加。
* named\_pipe\_full\_access\_group。被授予完全访问命名管道的Windows组的名称。在MySQL 8.0.14中添加。
* no-dd-upgrade。防止在启动时自动升级数据字典表。在MySQL 8.0.4中添加。
* no-monitor。不分叉RESTART所需的监控进程。在MySQL 8.0.12中添加。
* original\_commit\_timestamp。在原始源上提交事务的时间。在MySQL 8.0.1中添加。
* original\_server\_version。事务最初提交的服务器的MySQL服务器版本号。在MySQL 8.0.14中添加。
* partial\_revokes。是否启用了部分撤销。在MySQL 8.0.16中添加。
* password\_history。密码重用前所需的密码修改次数。在MySQL 8.0.3中添加。
* password\_require\_current。更改密码是否需要当前密码验证。在MySQL 8.0.13中添加。
* password\_reuse\_interval。密码重用前需要经过的天数。在MySQL 8.0.3中添加。
* performance\_schema\_max\_digest\_sample\_age。查询重新采样的年龄，以秒为单位。在MySQL 8.0.3中添加。
* performance\_schema\_show\_processlist。选择SHOW PROCESSLIST实现。在MySQL 8.0.22中添加。
* persist\_only\_admin\_x509\_subject。SSL证书X.509主题，可使持久化限制的系统变量持久化。在MySQL 8.0.14中添加。
* persisted\_globals\_load。是否加载持久化的配置设置。在MySQL 8.0.0中添加。
* print\_identified\_with\_as\_hex。对于SHOW CREATE USER，打印包含不可打印字符的十六进制哈希值。在MySQL 8.0.17中添加。
* protocol\_compression\_algorithms。为传入的连接允许的压缩算法。在MySQL 8.0.18中添加。
* regexp\_stack\_limit。正则表达式匹配堆栈大小限制。在MySQL 8.0.4中添加。
* regexp\_time\_limit。正则表达式匹配超时。在MySQL 8.0.4中添加。
* replication\_optimize\_for\_static\_plugin\_config。用于半同步复制的共享锁。在MySQL 8.0.23中添加。
* replication\_sender\_observe\_commit\_only。用于半同步复制的有限回调。在MySQL 8.0.23中添加。
* require\_row\_format。供内部服务器使用。在MySQL 8.0.19中添加。
* resultet\_metadata。服务器是否返回结果集元数据。在MySQL 8.0.3中添加。
* rpl\_read\_size。设置从二进制日志文件和中继日志文件读取的最小数据量（字节）。在MySQL 8.0.11中添加。
* secondary\_engine\_cost\_threshold。优化器成本阈值，用于将查询卸载到辅助引擎。在MySQL 8.0.16中添加。
* select\_into\_buffer\_size。用于OUTFILE或DUMPFILE导出文件的缓冲区的大小；覆盖read\_buffer\_size。在MySQL 8.0.22中添加。
* select\_into\_disk\_sync。在为OUTFILE或DUMPFILE导出文件刷新缓冲区后与存储设备同步数据；OFF禁用同步，是默认值。在MySQL 8.0.22中添加。
* select\_into\_disk\_sync\_delay。当select\_into\_sync\_disk = ON时，设置每次同步OUTFILE或DUMPFILE导出文件缓冲区后的延迟（毫秒），否则没有影响。在MySQL 8.0.22中添加。
* show\_create\_table\_skip\_secondary\_engine。是否从SHOW CREATE TABLE输出中排除SECONDARY ENGINE子句。在MySQL 8.0.18中添加。
* show\_create\_table\_verbosity。是否在SHOW CREATE TABLE中显示ROW\_FORMAT，即使它有默认值。在MySQL 8.0.11中添加。
* sql\_require\_primary\_key。表是否必须有主键。在MySQL 8.0.13中添加。
* ssl\_fips\_mode。是否在服务器端启用FIPS模式。在MySQL 8.0.11中添加。
* syseventlog.设施。用于系统日志消息的设施。在MySQL 8.0.13中添加。
* syseventlog.include\_pid。是否在syslog消息中包括服务器PID。在MySQL 8.0.13中添加。
* syseventlog.tag。syseventlog.tag：syslog消息中服务器标识符的标签。在MySQL 8.0.13中添加。
* table\_encryption\_privilege\_check。启用TABLE\_ENCRYPTION\_ADMIN权限检查。在MySQL 8.0.16中添加。
* temptable\_max\_mmap。TempTable存储引擎可以从内存映射的临时文件分配的最大内存量。在MySQL 8.0.23中添加。
* temptable\_max\_ram。定义在数据存储到磁盘之前，TempTable存储引擎可以占用的最大内存量。在MySQL 8.0.2中添加。
* temptable\_use\_mmap: 定义当达到temptable\_max\_ram阈值时，TempTable存储引擎是否分配内存映射的文件。在MySQL 8.0.16中添加。
* thread\_pool\_algorithm。线程池算法。在MySQL 8.0.11中添加。
* thread\_pool\_high\_priority\_connection。当前会话是否是高优先级。在MySQL 8.0.11中添加。
* thread\_pool\_max\_active\_query\_threads。每个组允许的最大活动查询线程数。在MySQL 8.0.19中添加。
* thread\_pool\_max\_unused\_threads。允许的未使用线程的最大数量。在MySQL 8.0.11中添加。
* thread\_pool\_prio\_kickup\_timer：语句被移到高优先级执行之前的时间。在MySQL 8.0.11中添加。
* thread\_pool\_size。线程池中线程组的数量。在MySQL 8.0.11中添加。
* thread\_pool\_stall\_limit：语句被定义为停滞前的时间。在MySQL 8.0.11中添加。
* tls\_ciphersuites。允许的用于加密连接的TLSv1.3密码套件。在MySQL 8.0.16中添加。
* 升级。控制启动时的自动升级。在MySQL 8.0.16中添加。
* use\_secondary\_engine。是否使用辅助引擎执行查询。在MySQL 8.0.13中添加。
* validate-config。验证服务器配置。在MySQL 8.0.16中添加。
* validate\_password.check\_user\_name：是否根据用户名检查密码。在MySQL 8.0.4中添加。
* validate\_password.dictionary\_file：验证密码的字典文件。在MySQL 8.0.4中添加。
* validate\_password.dictionary\_file\_last\_parsed：字典文件最后被解析的时间。在MySQL 8.0.4中添加。
* validate\_password.dictionary\_file\_words\_count: 词典文件中的字数。在MySQL 8.0.4中添加。
* validate\_password.length: validate\_password要求的密码长度。在MySQL 8.0.4中添加。
* validate\_password.mixed\_case\_count：validate\_password要求的大写/小写字符数。在MySQL 8.0.4中添加。
* validate\_password.number\_count：validate\_password需要的数字字符数量。在MySQL 8.0.4中添加。
* validate\_password.policy：validate\_password密码策略。在MySQL 8.0.4中添加。
* validate\_password.special\_char\_count：validate\_password要求的特殊字符数。在MySQL 8.0.4中添加。
* version\_compile\_zlib。编译的zlib库的版本。在MySQL 8.0.11中添加。
* windowing\_use\_high\_precision。是否以高精确度计算窗口函数。在MySQL 8.0.2中添加

**在MySQL 8.0中废弃的选项和变量**

下列系统变量、状态变量和选项在MySQL 8.0中已被废弃。

压缩。客户端连接是否在客户端/服务器协议中使用压缩。在MySQL 8.0.18中已废弃。

* admin-ssl：启用连接加密。在MySQL 8.0.25中被弃用。
* expire\_logs\_days。在这么多天后清除二进制日志。在MySQL 8.0.3中被弃用。
* group\_replication\_ip\_whitelist。允许连接到组的主机列表。在MySQL 8.0.22中被废弃。
* have\_openssl：mysqld是否支持SSL连接。在MySQL 8.0.25中被弃用。
* have\_ssl：mysqld是否支持SSL连接。在MySQL 8.0.25中被弃用。
* innodb\_undo\_tablespaces。回滚段划分的表空间文件的数量。在MySQL 8.0.4中被弃用。
* log\_bin\_use\_v1\_row\_events。服务器是否正在使用版本1的二进制日志行事件。在MySQL 8.0.18中被废弃。
* log\_syslog。是否将错误日志写到syslog。在MySQL 8.0.2中已废弃。
* master-info-file。在源的二进制日志中记住源和I/O复制线程的位置和文件名。在MySQL 8.0.18中被弃用。
* master\_info\_repository。是否将包含源信息和源的二进制日志中复制I/O线程位置的连接元数据库写入文件或表。在MySQL 8.0.23中被废弃。
* max\_length\_for\_sort\_data。排序记录中的最大字节数。在MySQL 8.0.20中已废弃。
* no-dd-upgrade。防止在启动时自动升级数据字典表。在MySQL 8.0.16中已废弃。
* relay\_log\_info\_file。应用者元数据存储库的文件名，复制器在其中记录有关中继日志的信息。在MySQL 8.0.18中被弃用。
* relay\_log\_info\_repository。是否将中继日志中的复制SQL线程的位置写入文件或表。在MySQL 8.0.23中已废弃。
* slave\_compressed\_protocol。使用源/复制协议的压缩。在MySQL 8.0.18中已废弃。
* slave\_rows\_search\_algorithms。决定用于复制体更新批处理的搜索算法。从这个列表中任选2或3个。index\_search, table\_scan, hash\_scan。在MySQL 8.0.18中被弃用。
* ssl：启用连接加密。在MySQL 8.0.25中被弃用。
* 符号链接（symbolic-links）。允许MyISAM表的符号链接。在MySQL 8.0.2中被弃用。
* temptable\_use\_mmap。定义当达到temptable\_max\_ram阈值时，TempTable存储引擎是否分配内存映射的文件。在MySQL 8.0.25中被弃用。

**在MySQL 8.0中删除的选项和变量**

以下系统变量、状态变量和选项已在MySQL 8.0中被删除。

* Com\_alter\_db\_upgrade。ALTER DATABASE ...的计数
* UPGRADE DATA DIRECTORY NAME语句。在MySQL 8.0.0中已删除。
* nnodb\_available\_undo\_logs。InnoDB回滚段的总数；与innodb\_rollback\_segments不同，后者显示活动回滚段的数量。在MySQL 8.0.2中被删除。
* Qcache\_free\_blocks。查询缓存中可用的内存块的数量。在MySQL 8.0.3中被删除。
* Qcache\_free\_memory。用于查询缓存的可用内存量。在MySQL 8.0.3中被删除。
* Qcache\_hits。查询缓存的命中率。在MySQL 8.0.3中被删除。
* Qcache\_inserts。查询缓存插入的数量。在MySQL 8.0.3中被删除。
* Qcache\_lowmem\_prunes。由于缓存缺乏可用内存而从查询缓存中删除的查询数量。在MySQL 8.0.3中被删除。
* Qcache\_not\_cached。非缓存查询的数量（不可缓存，或由于query\_cache\_type设置而未缓存）。在MySQL 8.0.3中被删除。
* Qcache\_queries\_in\_cache。在查询缓存中注册的查询数量。在MySQL 8.0.3中被删除。
* Qcache\_total\_blocks。查询缓存中的块的总数。在MySQL 8.0.3中被删除。
* Slave\_heartbeat\_period。复制的心跳间隔，单位是秒。在MySQL 8.0.1中被删除。
* Slave\_last\_heartbeat。显示何时收到最新的心跳信号，格式为TIMESTAMP。在MySQL 8.0.1中被删除。
* Slave\_received\_heartbeats。自上次重置以来复制体收到的心跳次数。在MySQL 8.0.1中被删除。
* Slave\_retried\_transactions。自启动以来，复制的SQL线程重试事务的总次数。在MySQL 8.0.1中被删除。
* Slave\_running。该服务器作为副本的状态（复制I/O线程状态）。在MySQL 8.0.1中被删除。
* bootstrap：由mysql安装脚本使用。在MySQL 8.0.0中被删除。
* date\_format。DATE格式（未使用）。在MySQL 8.0.3中删除。
* datetime\_format。DATETIME/TIMESTAMP格式（未使用）。在MySQL 8.0.3中被删除。
* des-key-file。从给定文件加载用于des\_encrypt()和des\_encrypt的密钥。在MySQL 8.0.3中被删除。
* group\_replication\_allow\_local\_disjoint\_gtids\_join。允许当前服务器加入组，即使它有不存在于组中的事务。在MySQL 8.0.4中被删除。
* have\_crypt。有crypt()系统调用。在MySQL 8.0.3中被删除。
* ignore-db-dir。将目录视为非数据库目录。在MySQL 8.0.0中删除。
* ignore\_builtin\_innodb。忽略内置InnoDB。在MySQL 8.0.3中删除。
* ignore\_db\_dirs。将目录视为非数据库目录。在MySQL 8.0.0中删除。
* innodb\_checksums。启用InnoDB校验和验证。在MySQL 8.0.0中删除。
* innodb\_disable\_resize\_buffer\_pool\_debug。禁用InnoDB缓冲池的大小调整。在MySQL 8.0.0中被删除。
* innodb\_file\_format。新InnoDB表的格式。在MySQL 8.0.0中被移除。
* innodb\_file\_format\_check。InnoDB是否执行文件格式兼容性检查。在MySQL 8.0.0中被删除。
* innodb\_file\_format\_max。共享表空间中的文件格式标签。在MySQL 8.0.0中被删除。
* innodb\_large\_prefix。启用列前缀索引的长键。在MySQL 8.0.0中被移除。
* innodb\_locks\_unsafe\_for\_binlog。强制InnoDB不使用下一个键的锁定。相反，只使用行级锁。在MySQL 8.0.0中被移除。
* innodb\_scan\_directories。定义了在InnoDB恢复期间扫描表空间文件的目录。在MySQL 8.0.4中被删除。
* innodb\_stats\_sample\_pages。为索引分布统计取样的索引页的数量。在MySQL 8.0.0中被删除。
* innodb\_support\_xa。启用InnoDB对XA两阶段提交的支持。在MySQL 8.0.0中被删除。
* innodb\_undo\_logs。InnoDB使用的Undo logs（回滚段）的数量；innodb\_rollback\_segments的别名。在MySQL 8.0.2中被移除。
* internal\_tmp\_disk\_storage\_engine：内部临时表的存储引擎。在MySQL 8.0.16中被删除。
* log-warnings。将一些非关键性的警告写入日志文件。在MySQL 8.0.3中删除。
* log\_builtin\_as\_identified\_by\_password：是否以向后兼容的方式记录CREATE/ALTER USER、GRANT。在MySQL 8.0.11中被删除。
* log\_error\_filter\_rules。用于错误记录的过滤规则。在MySQL 8.0.4中被删除。
* log\_syslog。是否将错误日志写到syslog。在MySQL 8.0.13中删除。
* log\_syslog\_facility。syslog消息的设施。在MySQL 8.0.13中被删除。
* log\_syslog\_include\_pid。是否在syslog消息中包括服务器PID。在MySQL 8.0.13中被删除。
* log\_syslog\_tag。syslog消息中服务器标识符的标签。在MySQL 8.0.13中被删除。
* max\_tmp\_tables。未使用。在MySQL 8.0.3中被删除。
* metadata\_locks\_cache\_size。元数据锁缓存的大小。在MySQL 8.0.13中删除。
* metadata\_locks\_hash\_instances。元数据锁哈希值的数量。在MySQL 8.0.13中被删除。
* multi\_range\_count。在范围选择时一次发送给表处理程序的最大范围数。在MySQL 8.0.3中删除。
* old\_passwords。为PASSWORD()选择密码散列方法。在MySQL 8.0.11中被删除。
* partition（分区）。启用（或禁用）分区支持。在MySQL 8.0.0中被删除。
* query\_cache\_limit：不缓存大于此值的结果。在MySQL 8.0.3中删除。
* query\_cache\_min\_res\_unit。分配结果空间的最小单位大小（最后一个单位在写入所有结果数据后被修剪）。在MySQL 8.0.3中被删除。
* query\_cache\_size。分配用于存储旧查询结果的内存。在MySQL 8.0.3中被删除。
* query\_cache\_type。查询缓存类型。在MySQL 8.0.3中删除。
* query\_cache\_wlock\_invalidate：在LOCK上写的查询缓存中无效的查询。在MySQL 8.0.3中删除。
* secure\_auth。不允许对有旧（4.1前）密码的账户进行认证。在MySQL 8.0.3中删除。
* show\_compatibility\_56：SHOW STATUS/VARIABLES的兼容性。在MySQL 8.0.1中删除。
* skip-partition。不启用用户定义的分区。在MySQL 8.0.0中删除。
* sync\_frm。创建时将.frm同步到磁盘。默认情况下是启用的。在MySQL 8.0.0中删除。
* temp-pool。使用该选项使创建的大多数临时文件使用一组小名称，而不是为每个新文件使用唯一的名称。在MySQL 8.0.1中被删除。
* time\_format。时间格式（未使用）。在MySQL 8.0.3中删除。
* tx\_isolation。默认的事务隔离级别。在MySQL 8.0.3中被删除。
* tx\_read\_only。默认的事务访问模式。在MySQL 8.0.3中被删除

## 1.5 MySQL的信息来源

本节列出了你可能会发现有帮助的其他信息来源，如MySQL网站、邮件列表、用户论坛和Internet Relay Chat。

**MySQL网站**

MySQL文档的主要网站是https://dev.mysql.com/doc/。在线和可下载的文档格式可用于MySQL参考手册、MySQL连接器等。

MySQL开发人员以MySQL服务器博客的形式提供有关新的和即将推出的功能的信息。

**MySQL论坛上的MySQL社区支持**

http://forums.mysql.com 的论坛是一个重要的社区资源。许多论坛都是可用的，分为以下几个一般类别。

* 迁移
* MySQL的使用
* MySQL连接器
* 编程语言
* 工具类
* 第三方应用程序
* 存储引擎
* 编码技术
* SQL标准
* 业务

**MySQL企业版**

Oracle以MySQL企业版的形式提供技术支持。对于那些在关键业务生产应用中依赖MySQL数据库管理系统的组织来说，MySQL企业版是一种商业订阅产品，其中包括。

* MySQL企业服务器
* MySQL企业监控器
* 每月快速更新和季度服务包
* MySQL知识库
* 24x7技术和咨询支持

MySQL企业版有多个层级，让您可以灵活地选择最符合您需求的服务级别。欲了解更多信息，请参阅MySQL企业版。

## 1.6 如何报告缺陷或问题

在发布关于一个问题的错误报告之前，请尝试验证它是否是一个错误，以及它是否已经被报告过。

* 首先在https://dev.mysql.com/doc/ 上搜索MySQL在线手册。我们试图通过经常更新手册以解决新发现的问题来保持手册的更新性。此外，手册所附的发行说明可能特别有用，因为很可能一个较新的版本包含对你问题的解决方案。发行说明可以在刚才给出的手册的位置上找到。
* 如果你得到一个SQL语句的解析错误，请仔细检查你的语法。如果你不能发现其中的问题，那么极有可能是你当前版本的MySQL Server不支持你所使用的语法。如果你使用的是当前版本，而手册没有涵盖你所使用的语法，那么MySQL Server不支持你的语句。

如果手册涵盖了你正在使用的语法，但你有一个较早的MySQL Server版本，你应该检查MySQL的变化历史，看看该语法是何时实现的。在这种情况下，你可以选择升级到较新版本的MySQL Server。

* 关于一些常见问题的解决方案，见B.3节 "问题和常见错误"。
* 在http://bugs.mysql.com/，搜索bug数据库，看看该bug是否已经被报告和修复。
* 你还可以使用http://www.mysql.com/search/，搜索位于MySQL网站的所有网页（包括手册）。

如果你在手册、错误数据库或邮件列表档案中找不到答案，请向你当地的MySQL专家查询。如果你仍然找不到你的问题的答案，请使用以下准则来报告错误。

报告bug的正常方法是访问http://bugs.mysql.com/，这是我们bug数据库的地址。这个数据库是公开的，任何人都可以浏览和搜索。如果你登录到该系统，你可以输入新的报告。

在错误数据库中发布的错误（http://bugs.mysql.com/），如果在某个特定的版本中得到纠正，会在发布说明中注明。

如果你发现MySQL服务器的安全漏洞，请立即通过发送电子邮件到<secalert\_us@oracle.com>让我们知道。例外。支持客户应将所有问题，包括安全漏洞，报告给Oracle支持部门：http://support.oracle.com/。

要与其他用户讨论问题，你可以使用MySQL社区Slack。

写一份好的错误报告需要耐心，但第一次就把它写对了，可以为我们和你自己节省时间。一份好的bug报告，包含bug的完整测试案例，使得我们很可能在下一个版本中修复这个bug。本节将帮助您正确地写出您的报告，这样您就不会浪费您的时间去做那些可能对我们没有什么帮助或根本没有帮助的事情。请仔细阅读本节，并确保在你的报告中包含所有在此描述的信息。

最好在发布之前，你应该使用最新的生产或开发版本的MySQL服务器来测试这个问题。任何人只要在你的测试案例上使用mysql test < script\_file，或者运行你在bug报告中包含的shell或Perl脚本，就应该能够重复这个bug。任何我们能够重复的错误都有很大的机会在下一个MySQL版本中被修复。

如果在bug报告中包括对问题的良好描述，这是最有帮助的。也就是说，举出一个很好的例子，说明你所做的一切导致了这个问题，并准确详细地描述问题本身。最好的报告是那些包括一个完整的例子，显示如何重现该错误或问题。见第5.9节，"调试MySQL"。

请记住，我们有可能对包含太多信息的报告作出回应，但对包含太少信息的报告则不会。人们经常省略事实，因为他们认为他们知道问题的原因，并认为一些细节并不重要。一个好的原则是，如果你对陈述某事有疑问，就陈述它。如果我们必须要求你提供最初报告中缺少的信息，那么在报告中多写几行比等待答案更快、更省事。

错误报告中最常见的错误是：(a)不包括你所使用的MySQL发行版的版本号，以及(b)没有完全描述MySQL服务器所安装的平台（包括平台类型和版本号）。这些都是高度相关的信息，在100个案例中的99个案例中，没有这些信息，错误报告就没有用。我们经常收到这样的问题："为什么这个东西对我不起作用？" 然后我们发现，要求的功能在那个MySQL版本中没有实现，或者报告中描述的错误在较新的MySQL版本中已经被修复。错误往往是与平台有关的。在这种情况下，如果不知道操作系统和平台的版本号，我们几乎不可能修复什么。

如果你从源代码编译MySQL，记得也要提供关于你的编译器的信息，如果它与问题有关。通常人们在编译器中发现bug并认为问题与MySQL有关。大多数编译器一直在开发中，并且一个版本一个版本地变得更好。为了确定你的问题是否与你的编译器有关，我们需要知道你使用的是什么编译器。请注意，每一个编译问题都应被视为一个错误，并相应地报告。

如果一个程序产生了一个错误信息，在你的报告中包括这个信息是非常重要的。如果我们试图从档案中搜索一些东西，最好是报告的错误信息与程序产生的信息完全一致。(甚至字母大小写也要注意。)最好是复制并粘贴整个错误信息到你的报告中。你不应该试图从记忆中重现该信息。

如果你在Connector/ODBC（MyODBC）上遇到问题，请尝试生成一个跟踪文件并与你的报告一起发送。参见如何报告Connector/ODBC的问题或错误。

如果你的报告包括你用mysql命令行工具运行的测试案例的长查询输出行，你可以通过使用--垂直选项或\G语句结束符使输出更可读。本节后面的EXPLAIN SELECT例子演示了\G的使用。

请在你的报告中包括以下信息。

* 你所使用的MySQL分布的版本号（例如，MySQL 5.7.10）。你可以通过执行mysqladmin版本找到你正在运行的版本。mysqladmin程序可以在你的MySQL安装目录下的bin目录中找到。
* 你遇到问题的机器的制造商和型号。
* 操作系统的名称和版本。如果你使用Windows，你通常可以通过双击 "我的电脑 "图标并拉下 "帮助/关于Windows "菜单来获得名称和版本号。对于大多数类似Unix的操作系统，你可以通过执行命令uname -a来获得这些信息。
* 有时，内存的数量（真实和虚拟）是相关的。如果有疑问，请包括这些值。
* 你的MySQL安装中的docs/INFO\_BIN文件的内容。这个文件包含关于MySQL如何配置和编译的信息。
* 如果你使用的是MySQL软件的源码分布，包括你使用的编译器的名称和版本号。如果你有一个二进制分布，包括分布名称。
* 如果问题发生在编译期间，包括确切的错误信息，也包括发生错误的文件中违规代码周围的几行内容。
* 如果mysqld死亡，你也应该报告导致mysqld意外退出的语句。你通常可以通过在启用查询日志的情况下运行mysqld，然后在mysqld退出后查看日志来获得这些信息。见第5.9节，"调试MySQL"。
* 如果一个数据库表与问题有关，在错误报告中包括SHOW CREATE TABLE db\_name.tbl\_name语句的输出。这是一个非常简单的方法来获得数据库中任何表的定义。这些信息有助于我们创建一个与你所经历的情况相匹配的情况。
* 问题发生时有效的SQL模式可能很重要，所以请报告sql\_mode系统变量的值。对于存储过程、存储函数和触发器对象，相关的sql\_mode值是对象被创建时的有效值。对于存储过程或函数，SHOW CREATE PROCEDURE或SHOW CREATE FUNCTION语句会显示相关的SQL模式，或者你可以查询INFORMATION\_SCHEMA获得相关信息。

SELECT ROUTINE\_SCHEMA, ROUTINE\_NAME, SQL\_MODE

FROM INFORMATION\_SCHEMA.ROUTINES;

对于触发器，你可以使用这个语句。

SELECT EVENT\_OBJECT\_SCHEMA, EVENT\_OBJECT\_TABLE, TRIGGER\_NAME, SQL\_MODE

FROM INFORMATION\_SCHEMA.TRIGGERS;

* 对于与性能有关的错误或SELECT语句的问题，你应该总是包括EXPLAIN SELECT ...的输出，以及至少SELECT语句产生的行数。你还应该包括每个涉及到的表的SHOW CREATE TABLE tbl\_name的输出。你提供的关于你的情况的信息越多，就越有可能有人能帮助你。

下面是一个非常好的错误报告的例子。这些语句是使用mysql命令行工具运行的。注意对于那些否则会提供非常长的输出行而难以阅读的语句，使用了\G语句终止符。

mysql> **SHOW VARIABLES;**

mysql> **SHOW COLUMNS FROM ...\G**

***<output from SHOW COLUMNS>***

mysql> **EXPLAIN SELECT ...\G**

***<output from EXPLAIN>***

mysql> **FLUSH STATUS;**

mysql> **SELECT ...;**

***<A short version of the output from SELECT,***

***including the time taken to run the query>***

mysql> **SHOW STATUS;**

***<output from SHOW STATUS>***

* 如果在运行mysqld时出现错误或问题，尝试提供一个再现异常的输入脚本。这个脚本应该包括任何必要的源文件。脚本越能重现你的情况，就越好。如果你能做出一个可重现的测试案例，你应该把它上传到错误报告的附件中。
* 如果你不能提供一个脚本，你至少应该在你的报告中包括mysqladmin variables extended-status processlist的输出，以提供一些关于你的系统是如何执行的信息。
* 如果你不能产生一个只有几行的测试案例，或者测试表太大而不能包括在错误报告中（超过10行），你应该使用mysqldump转储你的表，并创建一个README文件来描述你的问题。使用tar和gzip或zip创建一个你的文件的压缩档案。在你为我们的bug数据库（http://bugs.mysql.com/）发起一个bug报告后，点击bug报告中的文件标签，以获得将存档上传到bug数据库的指示。
* 如果你认为MySQL服务器从一个语句中产生了一个奇怪的结果，不仅要包括结果，还要包括你对这个结果应该是什么的看法，以及描述你的看法的基础的解释。
* 当你提供一个问题的例子时，最好使用你实际情况中存在的表名、变量名等，而不是想出新的名字。问题可能与某个表或变量的名称有关。这些情况也许很少，但安全总比遗憾好。毕竟，对你来说，提供一个使用你的实际情况的例子应该更容易，而且对我们来说，这无论如何是更好的。如果你有不想让别人在错误报告中看到的数据，你可以使用前面描述的文件标签上传。如果这些信息确实是绝密的，而且你甚至不想让我们看到它，请继续使用其他名称提供一个例子，但请将此视为最后的选择。
* 如果可能的话，包括给相关程序的所有选项。例如，指出你在启动mysqld服务器时使用的选项，以及你用于运行任何MySQL客户端程序的选项。诸如mysqld和mysql等程序的选项，以及配置脚本的选项，往往是解决问题的关键，而且非常相关。把它们包括在内永远是个好主意。如果你的问题涉及到用Perl或PHP等语言编写的程序，请包括语言处理器的版本号，以及该程序使用的任何模块的版本。例如，如果你有一个使用DBI和DBD::mysql模块的Perl脚本，包括Perl、DBI和DBD::mysql的版本号。
* 如果你的问题与权限系统有关，请包括mysqladmin reload的输出，以及你在尝试连接时得到的所有错误信息。当你测试你的权限时，你应该执行mysqladmin reload版本并尝试用给你带来麻烦的程序进行连接。
* 如果你有一个错误的补丁，请包括它。但是，如果你没有提供一些必要的信息，例如显示你的补丁所修复的错误的测试案例，就不要认为这个补丁就是我们所需要的，或者我们可以使用它。我们可能会发现你的补丁有问题，或者我们可能根本不理解它。如果是这样，我们就不能使用它。
* 如果我们不能验证该补丁的确切目的，我们也不会使用它。测试案例在这里帮助我们。显示补丁可以处理所有可能发生的情况。如果我们发现一个边界情况（即使是罕见的情况），补丁将无法工作，它可能是无用的。
* 关于错误是什么，为什么会发生，或者它取决于什么的猜测通常是错误的。即使是MySQL团队也不能在不首先使用调试器来确定错误的真正原因的情况下猜测这些东西。
* 在你的错误报告中指出你已经检查了参考手册和邮件存档，以便其他人知道你已经尝试自己解决这个问题。
* 如果你的数据出现损坏，或者你在访问某个特定的表时出现错误，首先用CHECK TABLE检查你的表。如果该语句报告了任何错误。
  + InnoDB崩溃恢复机制在服务器被杀死后重新启动时处理清理工作，所以在典型的操作中，不需要 "修复 "表。如果你遇到InnoDB表的错误，重新启动服务器，看看问题是否持续存在，或者错误是否只影响了内存中的缓存数据。如果数据在磁盘上被破坏，考虑在启用innodb\_force\_recovery选项的情况下重新启动，这样你就可以转储受影响的表。
  + 对于非事务表，尝试用 REPAIR TABLE 或 myisamchk 来修复它们。见第5章，MySQL服务器管理。

如果你运行的是Windows，请使用SHOW VARIABLES LIKE 'lower\_case\_table\_names' 语句来验证lower\_case\_table\_names的值。这个变量影响服务器如何处理数据库和表名的大写字母。它对一个给定值的影响应该如第9.2.3节 "标识符大小写敏感度 "中所描述的。

* 如果你经常得到损坏的表，你应该尝试找出发生这种情况的时间和原因。在这种情况下，MySQL数据目录中的错误日志可能包含一些关于发生了什么的信息。这是名称中带有.err后缀的文件。）见第5.4.2节，"错误日志"。请在你的错误报告中包括该文件的任何相关信息。通常情况下，如果在更新过程中没有任何东西杀死它，mysqld不应该损坏一个表。如果你能找到mysqld死亡的原因，我们就会更容易为你提供问题的修复。参见B.3.1节，"如何确定导致问题的原因"。
* 如果可能的话，下载并安装最新版本的MySQL服务器，并检查它是否解决了你的问题。所有版本的MySQL软件都经过了彻底的测试，应该可以毫无问题地工作。我们相信要使一切尽可能向后兼容，你应该能够毫无困难地切换MySQL版本。见第2.1.2节，"安装哪个MySQL版本和分布"。

## 1.7 遵守MySQL标准

本节描述了MySQL与ANSI/ISO SQL标准的关系。MySQL服务器有许多对SQL标准的扩展，在这里你可以找到它们是什么以及如何使用它们。你还可以找到有关MySQL服务器缺失的功能的信息，以及如何解决一些差异的问题。

自1986年以来，SQL标准一直在发展，存在几个版本。在本手册中，"SQL-92 "是指1992年发布的标准。"SQL:1999"、"SQL:2003"、"SQL:2008 "和 "SQL:2011 "是指在相应年份发布的标准版本，最后一个是最新的版本。我们使用 "SQL标准 "或 "标准SQL "来指任何时候的SQL标准的当前版本。

我们对产品的主要目标之一是继续努力符合SQL标准，但不牺牲速度和可靠性。我们并不害怕增加对SQL的扩展或对非SQL功能的支持，如果这大大增加了MySQL服务器对大部分用户群的可用性。HANDLER接口就是这种策略的一个例子。见第13.2.4节，"HANDLER声明"。

我们继续支持事务型和非事务型数据库，以满足关键任务的24/7使用和大量Web或日志使用。

MySQL服务器最初被设计为在小型计算机系统上处理中等规模的数据库（1千万至1亿行，或每张表约100MB）。今天，MySQL服务器可以处理TB级的数据库。

我们的目标不是实时支持，尽管MySQL的复制能力提供了重要的功能。

MySQL支持ODBC 0至3.51级。

MySQL支持使用NDBCLUSTER存储引擎的高可用性数据库集群。参见第23章，MySQL NDB Cluster 8.0。

我们实现了XML功能，支持W3C XPath标准的大部分内容。参见第12.12节，"XML功能"。

MySQL支持RFC 7159所定义的本地JSON数据类型，并基于ECMAScript标准（ECMA-262）。见第11.5节，"JSON数据类型"。MySQL还实现了SQL:2016标准预发布草案所规定的SQL/JSON函数的一个子集；更多信息见第12.18节，"JSON函数"。

**选择SQL模式**

MySQL服务器可以在不同的SQL模式下运行，并且可以根据sql\_mode系统变量的值为不同的客户端应用这些模式。DBA可以设置全局SQL模式以匹配站点服务器的操作要求，而每个应用程序可以根据自己的要求设置会话SQL模式。

模式影响到MySQL支持的SQL语法和它所执行的数据验证检查。这使得在不同的环境中使用MySQL以及与其他数据库服务器一起使用MySQL更加容易。

关于设置SQL模式的更多信息，见第5.1.11节 "服务器SQL模式"。

**在ANSI模式下运行MySQL**

要在ANSI模式下运行MySQL服务器，用--ANSI选项启动mysqld。在ANSI模式下运行服务器与用下列选项启动它是一样的。

--transaction-isolation=SERIALIZABLE --sql-mode=ANSI

为了在运行时达到同样的效果，执行这两条语句。

SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SET GLOBAL sql\_mode = 'ANSI';

你可以看到，将sql\_mode系统变量设置为'ANSI'可以启用所有与ANSI模式有关的SQL模式选项，如下所示。

mysql> **SET GLOBAL sql\_mode='ANSI';**

mysql> **SELECT @@GLOBAL.sql\_mode;**

->'REAL\_AS\_FLOAT,PIPES\_AS\_CONCAT,ANSI\_QUOTES,IGNORE\_SPACE,ANSI'

用--ansi模式运行服务器与将SQL模式设置为'ANSI'不太一样，因为--ansi选项还设置了事务隔离级别。

### 1.7.1 MySQL对标准SQL的扩展

MySQL服务器支持一些你不可能在其他SQL DBMS中找到的扩展。请注意，如果你使用它们，你的代码很可能无法移植到其他SQL服务器。在某些情况下，你可以通过使用以下形式的注释来编写包含MySQL扩展的代码，但仍然可以移植。

/\*! ***MySQL-specific code*** \*/

在这种情况下，MySQL服务器解析并执行注释内的代码，就像它对其他SQL语句一样，但其他SQL服务器应该忽略扩展。例如，MySQL服务器识别以下语句中的STRAIGHT\_JOIN关键字，但其他服务器不应如此。

SELECT /\*! STRAIGHT\_JOIN \*/ col1 FROM table1,table2 WHERE ...

如果你在 ! 字符后添加了一个版本号，只有当MySQL版本大于或等于指定的版本号时，才会执行注释内的语法。以下注释中的KEY\_BLOCK\_SIZE子句仅由MySQL 5.1.10或更高版本的服务器执行。

CREATE TABLE t1(a INT, KEY (a)) /\*!50110 KEY\_BLOCK\_SIZE=1024 \*/;

下面的描述列出了MySQL的扩展，按类别组织。

* 磁盘上的数据组织

MySQL服务器将每个数据库映射到MySQL数据目录下的一个目录，并将数据库中的表映射到数据库目录中的文件名。因此，在文件名区分大小写的操作系统（如大多数Unix系统）上，数据库和表名在MySQL Server中是区分大小写的。见第9.2.3节，"标识符的大小写敏感性"。

* 一般的语言语法
  + 默认情况下，字符串可以用 "以及'"括起来。如果启用了ANSI\_QUOTES SQL模式，字符串只能用'括起来，服务器将用 "括起来的字符串解释为标识符。
  + \ 是字符串中的转义字符。
  + 在SQL语句中，你可以用db\_name.tbl\_name语法访问不同数据库的表。一些SQL服务器提供同样的功能，但称之为用户空间。MySQL服务器不支持表空间，如在这样的语句中使用。CREATE TABLE ralph.my\_table ... IN my\_tablespace。
* SQL语句的语法
  + ANALYZE TABLE, CHECK TABLE, OPTIMIZE TABLE, 和 REPAIR TABLE 语句
  + CREATE DATABASE, DROP DATABASE, 和ALTER DATABASE语句。参见第13.1.12节 "CREATE DATABASE语句"，第13.1.24节 "DROP DATABASE语句"，和第13.1.2节 "ALTER DATABASE语句"。
  + DO语句。
  + EXPLAIN SELECT，获得关于表如何被查询优化器处理的描述。
  + FLUSH和RESET语句。
  + SET语句。参见第13.7.6.1节，"变量分配的SET语法"。
  + SHOW语句。见第13.7.7节，"SHOW语句"。许多MySQL特定的SHOW语句产生的信息可以通过使用SELECT查询INFORMATION\_SCHEMA而以更标准的方式获得。参见第26章， INFORMATION\_SCHEMA表。
  + 使用LOAD DATA。在许多情况下，这种语法与Oracle LOAD DATA兼容。参见章节13.2.7, "LOAD DATA语句"。
  + 使用RENAME TABLE。参见第13.1.36节，"RENAME TABLE语句"。
  + 使用REPLACE而不是DELETE加INSERT。参见第13.2.9节，"REPLACE语句"。
  + 在ALTER TABLE语句中使用CHANGE col\_name, DROP col\_name, 或DROP INDEX, IGNORE或RENAME。在一个ALTER TABLE语句中使用多个ADD、ALTER、DROP或CHANGE子句。参见章节13.1.9, "ALTER TABLE语句"。
  + 在CREATE TABLE语句中使用索引名称、列的前缀索引、以及使用INDEX或KEY。参见章节13.1.20，"CREATE TABLE语句"。
  + 在CREATE TABLE中使用TEMPORARY或IF NOT EXISTS。
  + 在DROP TABLE和DROP DATABASE中使用IF EXISTS。
  + 用一个DROP TABLE语句删除多个表的能力。
  + UPDATE 和 DELETE 语句的 ORDER BY 和 LIMIT 子句。
  + INSERT INTO tbl\_name SET col\_name = ... 语法。
  + INSERT和REPLACE语句的DELAYED子句。
  + INSERT, REPLACE, DELETE, 和UPDATE语句的LOW\_PRIORITY子句。
  + 在SELECT语句中使用INTO OUTFILE或INTO DUMPFILE。参见章节13.2.10, "SELECT语句"。
  + 在SELECT语句中使用STRAIGHT\_JOIN或SQL\_SMALL\_RESULT等选项。
  + 你不需要在GROUP BY子句中命名所有选择的列。这对于一些非常特殊，但相当正常的查询来说，会有更好的性能。参见第12.20节，"聚合函数"。
  + 你可以用GROUP BY指定ASC和DESC，而不仅仅是用ORDER BY。
  + 能够在语句中用:=赋值操作符设置变量。参见第9.4节，"用户定义的变量"。

**数据类型**

* MEDIUMINT, SET, 和ENUM数据类型，以及各种BLOB和TEXT数据类型。
* AUTO\_INCREMENT、BINARY、NULL、UNSIGNED和ZEROFILL数据类型属性。

**函数和操作符**

* 为了使从其他SQL环境迁移过来的用户更容易，MySQL服务器支持许多函数的别名。例如，所有字符串函数都支持标准SQL语法和ODBC语法。
* MySQL服务器理解||和&&运算符是指逻辑OR和AND，就像在C编程语言中一样。在MySQL Server中，||和OR是同义词，正如&&和AND一样。由于这种漂亮的语法，MySQL Server不支持用于字符串连接的标准SQL ||操作符；使用CONCAT()代替。因为CONCAT()可以接受任何数量的参数，所以很容易将||操作符的使用转换为MySQL Server。
* 使用COUNT(DISTINCT value\_list)，其中value\_list有多于一个元素。
* 字符串比较默认是不区分大小写的，排序由当前字符集的排序决定，默认是utf8mb4。要执行大小写敏感的比较，你应该用BINARY属性声明你的列，或者使用BINARY转换，这将导致使用底层字符代码值而不是词法排序来进行比较。
* %操作符是MOD()的同义词。也就是说，N % M等同于MOD(N,M)。％是为C语言程序员和与PostgreSQL兼容而支持的
* =, <>, <=, <, >=, >, <<, >>, <=>, AND, OR, 或 LIKE 操作符可以在 SELECT 语句中的输出列列表（在 FROM 的左边）的表达式中使用。例如
* mysql> **SELECT col1=1 AND col2=2 FROM my\_table;**
* LAST\_INSERT\_ID()函数返回最近的AUTO\_INCREMENT值。参见12.16节 "信息函数"。
* 在数字值上允许使用LIKE。
* REGEXP和NOT REGEXP扩展正则表达式运算符。
* CONCAT()或CHAR()有一个参数或两个以上的参数。在MySQL服务器中，这些函数可以接受可变数量的参数）。
* BIT\_COUNT()、CASE、ELT()、FROM\_DAYS()、FORMAT()、IF()、MD5()、PERIOD\_ADD()、PERIOD\_DIFF()、TO\_DAYS()和WEEKDAY()函数。
* 使用TRIM()来修剪子字符串。标准SQL只支持删除单个字符。
* GROUP BY函数STD(), BIT\_OR(), BIT\_AND(), BIT\_XOR(), 和GROUP\_CONCAT()。参见第12.20节，"聚合函数"。

### 1.7.2 MySQL与标准SQL的区别

我们努力使MySQL服务器遵循ANSI SQL标准和ODBC SQL标准，但MySQL服务器在某些情况下执行的操作不同。

* 在MySQL和标准SQL的权限系统之间有几个区别。例如，在MySQL中，当你删除一个表时，该表的权限不会自动撤销。你必须明确发出REVOKE语句来撤销一个表的权限。更多信息，请参见第13.7.1.8节，"REVOKE语句"。
* CAST()函数不支持向REAL或BIGINT的转换。参见章节12.11, "转换函数和操作符"。

#### 1.7.2.1 SELECT INTO TABLE的区别

MySQL服务器不支持SELECT ... INTO TABLE Sybase SQL扩展。相反，MySQL服务器支持INSERT INTO ... SELECT标准SQL语法，这基本上是同样的事情。见第13.2.6.1节，"INSERT ... SELECT语句"。比如说

INSERT INTO tbl\_temp2 (fld\_id)

SELECT tbl\_temp1.fld\_order\_id

FROM tbl\_temp1 WHERE tbl\_temp1.fld\_order\_id > 100;

另外，你可以使用SELECT ... INTO OUTFILE或CREATE TABLE ... SELECT。

你可以使用SELECT ... INTO和用户定义的变量。同样的语法也可以在存储程序中使用游标和局部变量。参见章节13.2.10.1, "SELECT ... INTO语句"。

另外，你可以使用SELECT ... INTO OUTFILE或CREATE TABLE ... SELECT。

你可以使用SELECT ... INTO和用户定义的变量。同样的语法也可以在存储程序中使用游标和局部变量。参见章节13.2.10.1, "SELECT ... INTO语句"。

#### 1.7.2.2 UPDATE的区别

如果你在表达式中访问要更新的表中的列，UPDATE使用该列的当前值。下面语句中的第二个赋值将col2设置为当前（更新）的col1值，而不是原来的col1值。其结果是col1和col2具有相同的值。这种行为与标准SQL不同。

UPDATE t1 SET col1 = col1 + 1, col2 = col1;

#### 1.7.2.3 FOREIGN KEY约束的差异

MySQL对外键约束的实现在以下关键方面与SQL标准不同。

* 如果在父表中有几条具有相同引用键值的记录，InnoDB会执行外键检查，就像其他具有相同键值的父行不存在一样。例如，如果你定义了一个RESTRICT类型的约束，并且有一条子记录有几条父记录，InnoDB不允许删除任何一条父记录
* 如果ON UPDATE CASCADE或者ON UPDATE SET NULL循环更新它之前在同一级联中更新过的表，它的行为就像RESTRICT。这意味着你不能使用自我参照的ON UPDATE CASCADE或ON UPDATE SET NULL操作。这是为了防止级联更新导致的无限循环。另一方面，一个自我参照的ON DELETE SET NULL是可能的，正如一个自我参照的ON DELETE CASCADE一样。级联操作的嵌套深度不能超过15层。
* 在插入、删除或更新许多行的SQL语句中，外键约束（如唯一约束）被逐行检查。当执行外键检查时，InnoDB对它必须检查的子记录或父记录设置共享行级锁。MySQL立即检查外键约束；检查不会被推迟到事务提交。根据SQL标准，默认行为应该是延迟检查。也就是说，约束只有在整个SQL语句被处理后才被检查。这意味着不可能删除一个使用外键引用自己的行。
* 没有一个存储引擎，包括InnoDB，能够识别或执行参考完整性约束定义中使用的MATCH子句。使用显式的MATCH子句不会产生指定的效果，而且会导致ON DELETE和ON UPDATE子句被忽略。应该避免指定MATCH。
* SQL标准中的MATCH子句控制了当与被引用表中的主键比较时如何处理复合（多列）外键中的NULL值。MySQL基本上实现了MATCH SIMPLE定义的语义，它允许一个外键全部或部分为空。在这种情况下，包含这种外键的（子表）行可以被插入，即使它不匹配被引用的（父）表中的任何行。(可以使用触发器来实现其他语义)。
* 由于性能的原因，MySQL要求被引用的列要有索引。然而，MySQL并不强制要求被引用的列是UNIQUE或被声明为NOT NULL。
* 引用非UNIQUE键的FOREIGN KEY约束不是标准SQL，而是InnoDB的扩展。另一方面，NDB存储引擎要求在任何作为外键引用的列上有一个明确的唯一键（或主键）。
* 在UPDATE或DELETE CASCADE等操作中，对非唯一键或包含NULL值的键的外键引用的处理并没有很好的定义。建议你使用只引用UNIQUE（包括PRIMARY）和NOT NULL键的外键。
* MySQL解析但忽略了 "内联REFERENCES规范"（如SQL标准中所定义），其中引用被定义为列规范的一部分。MySQL只接受作为单独的FOREIGN KEY规范的一部分而指定的REFERENCES条款。对于不支持外键的存储引擎（如MyISAM），MySQL服务器解析并忽略外键规范。

关于外键约束的信息，请参见第13.1.20.5节，"FOREIGN KEY约束"。

#### 1.7.2.4 '--'作为注释的开始

标准SQL使用C语法/\*这是一个注释\*/进行注释，MySQL服务器也支持这种语法。MySQL还支持该语法的扩展，使MySQL特定的SQL能够嵌入注释中，如第9.7节 "注释 "中所述。

标准SQL使用"--"作为开始-注释序列。MySQL服务器使用#作为开始注释字符。MySQL服务器还支持 -- 注释样式的一个变体。也就是说， -- 开始注释序列后面必须有一个空格（或一个控制字符，如换行）。需要空格是为了防止自动生成的SQL查询出现问题，这些查询使用诸如下面的结构，在这里我们自动插入付款的值。

UPDATE account SET credit=credit-payment

考虑一下如果payment有一个负值，如-1，会发生什么。

UPDATE account SET credit=credit--1

credit--1在SQL中是一个有效的表达式，但是--被解释为注释的开始，表达式的一部分被丢弃了。其结果是一条具有完全不同含义的语句。

UPDATE account SET credit=credit

该语句根本没有产生任何值的变化。这说明，允许注释以--开头会产生严重的后果。

使用我们的实现要求在--后面有一个空格，以便在MySQL服务器中被识别为一个开始--注释序列。因此，credit--1是可以安全使用的。

另一个安全的特点是，mysql命令行客户端会忽略以--开头的行。

### 1.7.3 MySQL如何处理约束条件

MySQL使你既能处理允许回滚的事务性表，又能处理不允许回滚的非事务性表。正因为如此，约束处理在MySQL中与其他DBMS中有些不同。我们必须处理这样的情况：你在一个非事务表中插入或更新了大量的行，当发生错误时，这些变化不能回滚。

基本的理念是，MySQL服务器在解析要执行的语句时，会尝试对任何可以检测到的东西产生错误，并尝试从执行语句时发生的任何错误中恢复。我们在大多数情况下都会这样做，但还不是所有的情况。

当错误发生时，MySQL的选择是在中间停止语句，或者尽可能地从问题中恢复并继续。默认情况下，服务器遵循后一种路线。这意味着，例如，服务器可能会将无效的值胁迫到最接近的有效值。

有几个SQL模式的选项可以为处理坏的数据值以及在发生错误时是继续执行语句还是中止执行提供更大的控制。使用这些选项，你可以将MySQL服务器配置为以更传统的方式行事，与其他拒绝不正当输入的DBMS一样。SQL模式可以在服务器启动时全局设置，以影响所有客户端。单个客户端可以在运行时设置SQL模式，这使得每个客户端可以选择最适合其要求的行为。参见第5.1.11节，"服务器SQL模式"。

下面几节描述了MySQL服务器如何处理不同类型的约束。

#### 1.7.3.1 PRIMARY KEY和UNIQUE索引约束

通常，对于违反主键、唯一键或外键约束的数据更改语句（如INSERT或UPDATE），会发生错误。如果你使用的是InnoDB这样的事务性存储引擎，MySQL会自动回滚该语句。如果你使用非事务存储引擎，MySQL会在发生错误的行处停止处理语句，并留下未处理的任何剩余行。

MySQL支持INSERT、UPDATE等的IGNORE关键字。如果你使用它，MySQL将忽略主键或唯一键的违反，并继续处理下一行。参见你正在使用的语句的章节（第13.2.6节，"INSERT语句"，第13.2.13节，"UPDATE语句"，等等）。

你可以用mysql\_info()C API函数获得关于实际插入或更新的行数的信息。你也可以使用SHOW WARNINGS语句。参见mysql\_info()，以及第13.7.7.42节 "SHOW WARNINGS语句"。

InnoDB和NDB表支持外键。参见章节1.7.3.2, "FOREIGN KEY约束条件"。

#### 1.7.3.2 FOREIGN KEY限制条件

外键让你在不同的表中交叉引用相关的数据，而外键约束有助于保持这种分散的数据的一致性。

MySQL在CREATE TABLE和ALTER TABLE语句中支持ON UPDATE和ON DELETE外键引用。可用的引用动作是RESTRICT、CASCADE、SET NULL和NO ACTION（默认）。

SET DEFAULT也被MySQL服务器支持，但目前被InnoDB拒绝为无效。由于MySQL不支持延迟约束检查，NO ACTION被视为RESTRICT。关于MySQL支持的外键的确切语法，见第13.1.20.5节，"FOREIGN KEY约束"。

允许使用MATCH FULL、MATCH PARTIAL和MATCH SIMPLE，但应避免使用它们，因为它们导致MySQL服务器忽略同一语句中使用的任何ON DELETE或ON UPDATE子句。MATCH选项在MySQL中没有任何其他效果，它实际上是全时执行MATCH SIMPLE语义的。

MySQL要求外键列是有索引的；如果你创建一个有外键约束的表，但在一个给定的列上没有索引，就会创建一个索引。

你可以从INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE表中获得关于外键的信息。这里显示了一个针对这个表的查询的例子。

mysql> **SELECT TABLE\_SCHEMA, TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME**

> **FROM INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE**

> **WHERE REFERENCED\_TABLE\_SCHEMA IS NOT NULL;**

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

| TABLE\_SCHEMA | TABLE\_NAME | COLUMN\_NAME | CONSTRAINT\_NAME |

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

| fk1 | myuser | myuser\_id | f |

| fk1 | product\_order | customer\_id | f2 |

| fk1 | product\_order | product\_id | f1 |

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

3 rows in set (0.01 sec)

关于InnoDB表的外键信息也可以在INFORMATION\_SCHEMA数据库的INNODB\_FOREIGN和INNODB\_FOREIGN\_COLS表中找到。

InnoDB和NDB表支持外键。

#### 1.7.3.3 无效数据的强制约束

默认情况下，MySQL 8.0拒绝无效或不适当的数据值，并中止出现这些数据的语句。可以通过禁用严格的SQL模式（见第5.1.11节，"服务器SQL模式"）来改变这种行为，使其对无效值更加宽容，从而使服务器强制它们成为数据输入的有效值，但不建议这样做。

旧版本的MySQL默认采用宽容的行为；关于这种行为的描述，见无效数据的约束。

#### 1.7.3.4 ENUM和SET约束条件

ENUM和SET列提供了一种有效的方法来定义只能包含一组给定值的列。参见第11.3.5节 "ENUM类型 "和第11.3.6节 "SET类型"。

除非禁用严格模式（不推荐使用，但请参见第5.1.11节 "服务器SQL模式"），否则ENUM或SET列的定义对输入该列的值起到了约束作用。对于不满足这些条件的值会发生错误。

* ENUM值必须是列定义中列出的值之一，或者是其内部的数字等价物。该值不能是错误值（即0或空字符串）。对于一个定义为ENUM('a','b','c')的列，诸如'','d',或'ax'的值是无效的，会被拒绝。
* 一个SET值必须是空字符串或仅由列定义中列出的值组成，并以逗号分隔。对于一个定义为SET('a','b','c')的列，诸如'd'或'a,b,c,d'的值是无效的，会被拒绝。

如果你使用INSERT IGNORE或UPDATE IGNORE，无效值的错误可以在严格模式下被抑制。在这种情况下，会产生一个警告而不是一个错误。对于ENUM，该值被插入作为错误成员（0）。对于SET来说，除了删除任何无效的子串之外，值会按照给定的方式插入。例如，'a,x,b,y'的结果是一个'a,b'的值。

## **1.8 贡献者**

以下各节列出了帮助使MySQL成为今天这个样子的开发者、贡献者和支持者**。**

### **1.8.1 MySQL的贡献者**

尽管甲骨文公司和/或其附属机构拥有MySQL服务器和MySQL手册的所有版权，但我们希望承认那些对MySQL发行版作出了这样或那样贡献的人。贡献者被列在这里，顺序有点随机**。**

* Gianmassimo Vigazzola <qwerg@mbox.vol.it> 或 <qwerg@tin.it>

**最初移植到Win32/NT**

* **Per Eric Olsson**

**对于动态记录格式的建设性批评和实际测试。**

* **Irena Pancirov <irena@mail.yacc.it>**

**使用Borland编译器的Win32移植。 mysqlshutdown.exe和mysqlwatch.exe。**

* **David J. Hughes**

**为了努力做一个共享软件的SQL数据库。在TcX，即MySQL AB的前身，我们开始使用mSQL，但发现它不能满足我们的目的，所以我们改为为我们**的应用程序构建器Unireg写了一个SQ**L接口。mysqladmin和mysql client是主要受mSQL对应程序影响的程序。我们花了很多精力使MySQL的语法成为mSQL的超集。API的许多想法都是从mSQL借来的，以使免费的mSQL程序很容易移植到MySQL API上。MySQL软件不包含任何来自mSQL的代码。发行版中的两个文件（client/insert\_test.c和client/select\_test.c）是基于mSQL发行版中相应的（非版权）文件，但作为例子被修改，显示了将代码从mSQL转换到MySQL服务器所需的修改。(mSQL的版权归David J. Hughes所有。)**

* **Patrick Lynch**

**为帮助我们获得http://www.mysql.com/。**

* **Fred Lindberg**

**为设置qmail来处理MySQL邮件列表，以及为我们在管理MySQL邮件列表中得到的难以置信的帮助。**

* **Igor Romanenko <igor@frog.kiev.ua>**

**mysqldump（以前是msqldump，但由Monty移植和增强）。**

* **Yuri Dario**

**为了保持和扩展MySQL OS/2的移植。**

* **Tim Bunce**

**mysqlhotcopy的作者。**

* **Zarko Mocnik <zarko.mocnik@dem.si>**

**为斯洛文尼亚语进行排序。**

* **"TAMITO" <tommy@valley.ne.jp>**

**\_MB字符集宏以及ujis和sjis字符集。**

* **Joshua Chamas <joshua@chamas.com>**

**并发插入、扩展日期语法、在NT上调试的基础，以及在MySQL邮件列表上的回答。**

* **Yves Carlier <Yves.Carlier@rug.ac.be>**

**mysqlaccess，一个显示用户访问权限的程序。**

* **Rhys Jones <rhys@wales.com> (And GWE Technologies Limited)**

**用于早期的JDBC驱动之一。**

* **朱晓坤博士 <X.Zhu@brad.ac.uk>**

**进一步开发早期的JDBC驱动之一和其他与MySQL有关的Java工具。**

* **James Cooper <pixel@organic.com>**

**因为在他的网站上建立了一个可搜索的邮件列表档案。**

* **Rick Mehalick <Rick\_Mehalick@i-o.com>**

**用于xmysql，一个用于MySQL服务器的图形化X客户端。**

* **Doug Sisk <sisk@wix.com>**

**为Red Hat Linux提供MySQL的RPM包。**

* **Diemand Alexander V. <axeld@vial.ethz.ch>**

**为Red Hat Linux-Alpha提供MySQL的RPM包。**

* **Antoni Pamies Olive <toni@readysoft.es>**

**为英特尔和SPARC提供大量MySQL客户端的RPM版本。**

* **Jay Bloodworth <jay@pathways.sde.state.sc.us>**

**为MySQL 3.21提供了RPM版本。**

* **David Sacerdote <davids@secnet.com>**

**关于安全检查DNS主机名的想法。**

* **Wei-Jou Chen <jou@nematic.ieo.nctu.edu.tw>**

**对中文(BIG5)字符的一些支持。**

* **Wei He <hewei@mail.ied.ac.cn>**

**为中文(GBK)字符集提供了很多功能。**

* **Jan Pazdziora <adelton@fi.muni.cz>**

**捷克语的排序顺序。**

* **Zeev Suraski <bourbon@netvision.net.il>**

**FROM\_UNIXTIME()时间格式化，ENCRYPT()函数，以及bison顾问。活跃的邮件列表成员。**

* **Luuk de Boer <luuk@wxs.nl>**

**移植（和扩展）了基准套件到 DBI/DBD。对crash-me和运行基准有很大帮助。一些新的日期函数。mysql\_setpermission脚本。**

* **Alexis Mikhailov <root@medinf.chuvashia.su>**

**用户定义的函数（UDFs）；CREATE FUNCTION和DROP FUNCTION。**

* **Andreas F. Bobak <bobak@relog.ch>**

**AGGREGATE扩展到用户定义的函数。**

**Ross Wakelin <R.Wakelin@march.co.uk>**

**帮助设置InstallShield for MySQL-Win32。**

**Jethro Wright III <jetman@li.net>**

**libmysql.dll库。**

* **James Pereria <jpereira@iafrica.com>**

**Mysqlmanager，一个用于管理MySQL服务器的Win32 GUI工具。**

* **Curt Sampson <cjs@portal.ca>**

**将MIT-pthreads移植到NetBSD/Alpha和NetBSD 1.3/i386。**

* **Martin Ramsch <m.ramsch@computer.org>**

**MySQL教程中的例子。**

* **Steve Harvey**

**用于使mysqlaccess更加安全。**

**持久系统私人有限公司的Konark IA-64中心**

**对MySQL服务器的Win64端口的帮助。**

* **Albert Chin-A-Young.**

**为Tru64配置更新，大文件支持和更好的TCP包装器支持。**

* **John Birrell**

**对OS/2的pthread\_mutex()进行仿真。**

* **Benjamin Pflugmann**

**扩展MERGE表以处理INSERTS。是MySQL邮件列表中的活跃成员**

* **Jocelyn Fournier**

**优秀的发现和报告无数的bug（特别是在MySQL 4.1的子查询代码中）。**

### **1.8.2 文档作者和翻译者**

**以下人员帮助我们编写了MySQL文档和翻译了MySQL中的文档或错误信息。**

* **Kim Aldale**

**帮助将Monty和David的早期尝试改写成英文。**

* **Michael J. Miller Jr. <mke@terrapin.turbolift.com>**

**为第一本MySQL手册。以及为FAQ（很久以前变成了MySQL手册）做了大量的拼写/语言修正。**

* **严彩玲**

**2000年初将MySQL参考手册翻译成简体中文的第一人，Big5和香港的编码版本就是基于此。**

* **Jay Flaherty <fty@mediapulse.com>**

**手册中Perl DBI/DBD部分的大部头。**

* **Paul Southworth <pauls@etext.org>, Ray Loyzaga <yar@cs.su.oz.au>**

**对《参考手册》进行校对。**

* **Therrien Gilbert <gilbert@ican.net>, Jean-Marc Pouyot <jmp@scalaire.fr>**

**法语错误信息。**

* **Petr Snajdr, <snajdr@pvt.net>**

**捷克语错误信息。**

* **Jaroslaw Lewandowski <jotel@itnet.com.pl>**

**波兰语错误信息。**

* **Miguel Angel Fernandez Roiz**

**西班牙的错误信息。**

* **Roy-Magne Mo <rmo@www.hivolda.no>**

**挪威的错误信息和MySQL 3.21.xx的测试。**

* **Timur I. Bakeyev <root@timur.tatarstan.ru>**

**俄罗斯的错误信息。**

* **<brenno@dewinter.com> & Filippo Grassilli <phil@hyppo.com>**

**意大利语错误信息。**

* **Dirk Munzinger <dirk@trinity.saar.de>**

**德国的错误信息。**

* **Billik Stefan <billik@sun.uniag.sk>**

**斯洛伐克语的错误信息。**

* **Stefan Saroiu <tzoompy@cs.washington.edu>**

**罗马尼亚语的错误信息。**

* **Peter Feher**

**匈牙利语错误信息。**

* **Roberto M. Serqueira**

**葡萄牙语错误信息。**

* **Carsten H. Pedersen**

**丹麦语错误信息。**

* **Arjen Lentz**

**荷兰语错误信息，完成了早期的部分翻译（还包括一致性和拼写方面的工作）**

### **1.8.3 支持MySQL的软件包**

以下是一些最**重要的**API/软件包/应用程序的创建者/维护者的列表，很多人都在使用MySQL

我们不能在这里列出每一个可能的包，因为这样的列表就很难维护。对于其他软件包，请参考软件门户http://solutions.mysql.com/software/

* **Tim Bunce, Alligator Descartes**

**对于DBD（Perl）接口。**

* **Andreas Koenig <a.koenig@mind.de>**

**用于MySQL服务器的Perl接口。**

* **Jochen Wiedmann <wiedmann@neckar-alb.de>**

**用于维护Perl DBD::mysql模块。**

* **Eugene Chan <eugene@acenet.com.sg>**

**用于为MySQL服务器移植PHP。**

* **Georg Richter**

**负责MySQL 4.1的测试和bug查找。新的PHP 5.0 mysqli扩展（API）用于MySQL 4.1及以上版本。**

* **Giovanni Maruzzelli <maruzz@matrice.it>**

**用于移植iODBC（Unix ODBC）。**

* **Xavier Leroy <Xavier.Leroy@inria.fr>**

**LinuxThreads（由Linux上的MySQL服务器使用）的作者。**

### **1.8.4 用于创建MySQL的工具**

**下面是我们用于创建MySQL的一些工具的列表。我们用它来表达我们对创造这些工具的人的感谢，因为没有这些工具，我们不可能使MySQL成为今天这样。**

* **自由软件基金会**

**我们从他们那里得到了优秀的编译器（gcc）、优秀的调试器（gdb）和libc库（我们从它那里借用了strto.c来使一些代码在Linux中工作）。**

* **自由软件基金会和XEmacs开发团队**

**为了一个真正伟大的编辑器/环境。**

* **朱利安-苏厄德**

**valgrind的作者，一个优秀的内存检查工具，帮助我们发现了很多原本在MySQL中很难发现的bug。**

* **Dorothea Lütkehaus 和 Andreas Zeller**

**DDD（数据显示调试器）的作者，它是gdb的一个优秀的图形化前端）。**

### 1.8.5 MySQL的支持者

尽管Oracle公司和/或其附属机构拥有MySQL服务器和MySQL手册的所有版权，但我们希望承认以下公司，它们帮助我们资助了MySQL服务器的开发，例如为我们开发一个新的功能支付报酬，或为我们开发MySQL服务器提供硬件。

* VA Linux / Andover.net

资助了复制。

* NuSphere

编辑MySQL手册。

* Stork设计工作室

1998-2000年间使用的MySQL网站。

* 英特尔

为Windows和Linux平台的开发做出了贡献。

* 康柏公司

为在Linux/Alpha平台上的开发做出了贡献

* SWSoft

在嵌入式mysqld版本上的开发。

* FutureQuest

--skip-show-database选项