# InnoDB存储引擎

如果InnoDB不是您的默认存储引擎，则可以通过在命令行上使用[--default-storage-engine=InnoDB](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_default_storage_engine) 或在配置文件[default-storage-engine=innodb](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_default_storage_engine) 的[mysqld]部分中 定义重新启动服务器 来确定数据库服务器或应用程序是否正常工作my.cnf。

由于更改默认存储引擎只会在创建表时影响新表，请运行所有应用程序安装和设置步骤，以确认所有安装都正确。然后运行所有应用程序功能，以确保所有的数据加载，编辑和查询功能都可以工作。如果一个表依赖于一些MyISAM特定的功能，你会收到一个错误; 将语句添加ENGINE=MyISAM到 [CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html" \o "13.1.18 CREATE TABLE语法)语句以避免错误。

如果您没有对存储引擎做出故意的决定，并且您只想预览某些表在下创建时如何工作InnoDB，请[ALTER TABLE table\_name ENGINE=InnoDB;](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html" \o "13.1.8 ALTER TABLE语法)为每个表发出命令 。或者，为了运行测试查询和其他语句而不打扰原始表格，请制作如下所示的副本：

CREATE TABLE InnoDB\_Table （...） ENGINE=InnoDB AS SELECT \* FROM MyISAM\_Table;

要在真实的工作负载下，通过完整的应用程序获得真正的性能，请安装最新的MySQL服务器并运行基准测试。

测试完整的应用程序生命周期，从安装，大量使用和服务器重新启动。在数据库忙时模拟电源故障时，请停止服务器进程，并在重新启动服务器时验证数据是否已成功恢复。

测试任何复制配置，特别是如果您在主服务器和从属服务器上使用不同的MySQL版本和选项。

Oracle建议InnoDB作为典型数据库应用程序的首选存储引擎，从本地系统上运行的单用户维基和博客到推动性能极限的高端应用程序。在MySQL 5.7中，InnoDB是新表的默认存储引擎。

**重要**

InnoDB不能禁用。该 [--skip-innodb](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "option_mysqld_innodb) 选项已弃用，不起作用，其使用会导致警告。将在未来的MySQL版本中删除它。这也适用于它的同义词（--innodb=OFF， --disable-innodb等等）。

## 3、InnoDB多版本

InnoDB是一个 [多版本的存储引擎](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_mvcc" \o "MVCC)：它保留有关旧版本已更改行的信息，以支持事务功能，如并发和 [回滚](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_rollback" \o "回滚)。该信息存储在称为[回滚段](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_rollback_segment" \o "回滚段)（在Oracle中类似的数据结构之后）的数据结构中的表空间中 。InnoDB 使用回滚段中的信息执行事务回滚所需的撤消操作。它还使用这些信息来构建一行的较早版本以便 [一致性读取](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_consistent_read" \o "一致阅读)。

在内部，InnoDB为数据库中存储的每一行添加三个字段。6字节DB\_TRX\_ID字段指示插入或更新行的最后一个事务的事务标识符。此外，删除在内部被处理为更新，其中该行中的特殊位被设置为将其标记为已删除。每行还包含一个DB\_ROLL\_PTR称为滚动指针的7字节 字段。滚动指针指向写入回滚段的撤销日志记录。如果行已更新，则撤消日志记录包含在更新行之前重建该行的内容所需的信息。6字节DB\_ROW\_ID字段包含插入新行时单行增加的行ID。如果 InnoDB自动生成聚簇索引，则索引包含行ID值。 DB\_ROW\_ID

撤销回滚段中的日志分为插入和更新撤消日志。只有在事务回滚中才需要插入撤消日志，只要事务提交就可以将其删除。更新undo日志也用于一致的读取，但是只有在没有InnoDB分配了快照的事务之后，它们才能被丢弃 ，在一致的读取中可能需要更新undo log中的信息来构建数据库的早期版本行。

定期提交您的交易，包括仅发出一致性读取的交易。否则， InnoDB不能从更新撤销日志中丢弃数据，并且回滚段可能会增长太大，从而填满表空间。

回滚段中的撤消日志记录的物理大小通常小于相应的插入或更新行。您可以使用此信息来计算回滚段所需的空间。

在InnoDB多版本化方案中，当您使用SQL语句删除一行时，不会立即从数据库中物理删除一行。InnoDB只有当物理删除相应的行及其索引记录丢弃为删除写入的更新撤消日志记录时才会物理删除。这个删除操作被称为[清除](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_purge" \o "清除)，它是相当快的，通常采取与执行删除的SQL语句相同的时间顺序。

如果您在表中以相同的速率插入和删除小批量的行，则清除线程可能会开始滞后，并且由于所有“ 死 ”行，表可能会变得越来越大 ，从而使所有磁盘绑定和非常慢。在这种情况下，调节新行操作，并通过调整[innodb\_max\_purge\_lag](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "sysvar_innodb_max_purge_lag)系统变量将更多资源分配给清除线程 。有关详细信息[，](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \o "14.14 InnoDB启动选项和系统变量)请参见[第14.14节“InnoDB启动选项和系统变量”](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \o "14.14 InnoDB Startup Options and System Variables)。

多版本和二级索引

InnoDB多分支并发控制（MVCC）会将二次索引与聚簇索引不同。集群索引中的记录就地更新，并且其隐藏的系统列指向可以重建早期版本的记录的日志条目。与集群索引记录不同，辅助索引记录不包含隐藏的系统列，也不会在现场进行更新。

当二级索引列被更新时，旧的二级索引记录被删除标记，插入新记录，并且最终清除删除标记的记录。当二级索引记录被删除标记或辅助索引页由更新的事务更新时，InnoDB查找聚簇索引中的数据库记录。在聚簇索引中，DB\_TRX\_ID检查记录，如果记录在读取事务启动后被修改，则会从undo日志中检索正确的记录版本。

如果二次索引记录被标记为删除，或者次索引页由更新的事务更新， 则不使用[覆盖索引](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_covering_index" \o "覆盖指数)技术。不要从索引结构返回值，而是InnoDB查找聚簇索引中的记录。

然而，如果启用了 [索引条件下推（ICP）](https://dev.mysql.com/doc/refman/5.7/en/index-condition-pushdown-optimization.html" \o "8.2.1.5索引条件下推优化)优化，并且WHERE可以仅使用索引中的字段来评估部分条件，则MySQL服务器仍将该部分WHERE条件推送到使用指数。如果没有找到匹配的记录，则避免聚类索引查找。如果找到匹配的记录，即使在删除标记的记录中， InnoDB查找聚簇索引中的记录。

## 4、InnoDB架构

本小节主要介绍InnoDB的各种组件。

（8）撤销日志

（9）每个表一个表空间文件（File-Per-Table Tablespaces）

File-Per-Table模式开启后，每一张表都会有自己的数据文件（table\_name.ibd文件）而不是将表的数据放在系统表空间中。[innodb\_file\_per\_table](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "sysvar_innodb_file_per_table)选项启用该模式。每个表的数据文件在数据库目录中创建。File-Per-Table模式支持DYNAMIC和 COMPRESSED行格式，从而支持诸如将可变长度数据存储在页面外和表压缩等功能。它的更多特点和优点后边分析。

（10）通用表空间

InnoDB共享的通用表空间由[CREATE TABLESPACE](https://dev.mysql.com/doc/refman/5.7/en/create-tablespace.html" \o "13.1.19 CREATE TABLESPACE Syntax) 创建，通用表空间可以创建在MySQL数据目录外的其他目录中，可以包含多个表的内容，并且支持所有的行格式。

表可以使用如下语句添加到表空间中：

[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html" \o "13.1.18 CREATE TABLE Syntax)***[tbl\_name](https://dev.mysql.com/doc/refman/5.7/en/create-table.html" \o "13.1.18 CREATE TABLE Syntax)***[... TABLESPACE [=]](https://dev.mysql.com/doc/refman/5.7/en/create-table.html" \o "13.1.18 CREATE TABLE Syntax)***[tablespace\_name](https://dev.mysql.com/doc/refman/5.7/en/create-table.html" \o "13.1.18 CREATE TABLE Syntax)***

或者：

[ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html" \o "13.1.8 ALTER TABLE Syntax)***[tbl\_name](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html" \o "13.1.8 ALTER TABLE Syntax)***[TABLESPACE [=]](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html" \o "13.1.8 ALTER TABLE Syntax)***[tablespace\_name](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html" \o "13.1.8 ALTER TABLE Syntax)***

（11）undo表空间

默认情况下，undo log存储在系统表空间中，现在也可以将多个undo log存放在undo表空间（撤销表空间）中。undo表空间由[innodb\_undo\_tablespaces](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "sysvar_innodb_undo_tablespaces) 选项配置，如果该值为非0，那么MySQL启动时，InnoDB将会创建undo表空间。

（12）临时表空间

临时表空间用于非压缩 InnoDB临时表和相关对象。的[innodb\_temp\_data\_file\_path](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "sysvar_innodb_temp_data_file_path) 配置选项定义了临时表数据文件的相对路径。如果 [innodb\_temp\_data\_file\_path](https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html" \l "sysvar_innodb_temp_data_file_path)未定义，ibtmp1则在数据目录中创建一个自动扩展的12MB数据文件 。在每个服务器上重新创建临时表空间，并接收动态生成的空间ID，这有助于避免与现有空间ID的冲突。临时表空间不能驻留在原始设备上。如果无法创建临时表空间，则拒绝启动。

临时表空间在正常关机或中止初始化时被删除。发生崩溃时，临时表空间不会被删除。在这种情况下，数据库管理员可以手动删除临时表空间或重新启动具有相同配置的服务器，从而删除并重新创建临时表空间。

**临时表撤消日志：**

临时表撤销日志用于临时表和相关对象。此类型的[撤销日志](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_undo_log" \o "撤销日志)不是重做日志，因为临时表在崩溃恢复期间不会恢复，并且不需要重做日志。但是，临时表撤销日志用于服务器运行时的回滚。这种特殊类型的非重做撤销日志通过避免临时表和相关对象的重做日志记录I / O来提高性能。临时表撤销日志驻留在临时表空间中。32个 [回滚段](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_rollback_segment" \o "回滚段) 保留用于修改临时表和相关对象的事务的临时表 [撤销日志](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_undo_log" \o "撤销日志)。

（13）redo log（重做日志）

重做日志是在崩溃恢复期间使用的基于磁盘的数据结构，以纠正不完整事务写入的数据。在正常操作期间，重做日志编码要更改InnoDB由SQL语句或低级API调用导致的表数据的请求 。在初始化期间和连接被接受之前，会自动重播在意外关闭之前未完成更新数据文件的修改。有关重做日志在崩溃恢复中的作用的信息，请参见[第14.18.2节“InnoDB恢复”](https://dev.mysql.com/doc/refman/5.7/en/innodb-recovery.html" \o "14.18.2 InnoDB恢复)。

默认情况下，重做日志物理磁盘上的一组文件，命名为代表的ib\_logfile0和 ib\_logfile1。MySQL以循环方式写入重做日志文件。重做日志中的数据以影响的记录编码; 该数据统称为重做。通过重做日志的数据通过越来越多的[LSN](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_lsn" \o "LSN)值来表示。

有关相关信息，请参阅：

* [第14.6.1节“InnoDB启动配置”](https://dev.mysql.com/doc/refman/5.7/en/innodb-init-startup-configuration.html" \o "14.6.1 InnoDB启动配置)
* [第8.5.4节“优化InnoDB重做日志记录”](https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-logging.html" \o "8.5.4优化InnoDB重做日志记录)
* [第14.7.2节“更改InnoDB重做日志文件的数量或大小”](https://dev.mysql.com/doc/refman/5.7/en/innodb-data-log-reconfiguration.html" \o "14.7.2更改InnoDB重做日志文件的数量或大小)
* [InnoDB崩溃恢复](https://dev.mysql.com/doc/refman/5.7/en/innodb-recovery.html" \l "innodb-crash-recovery" \o "InnoDB崩溃恢复)

**重做日志刷新的组提交**

InnoDB像任何其他与 [ACID](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_acid" \o "酸)兼容的数据库引擎一样，在事务提交之前刷新[重做日志](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_redo_log" \o "重做日志)。InnoDB 使用[组提交](https://dev.mysql.com/doc/refman/5.7/en/glossary.html" \l "glos_group_commit" \o "组提交) 功能将多个这样的刷新请求组合在一起，以避免每次提交一次刷新。使用组提交时，会 InnoDB发出单次写入日志文件以对同时提交的多个用户事务执行提交操作，从而显着提高吞吐量。

有关性能COMMIT和其他事务操作的更多信息 ，请参见[第8.5.2节“优化InnoDB事务管理”](https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-transaction-management.html" \o "8.5.2优化InnoDB事务管理)。

## 5、InnoDB锁和事务模型

要实现大规模、繁忙或高度可靠的数据库应用程序，并调整MySQL的性能，那么了解InnoDB锁和InnoDB 事务模型将是非常重要的 。

（1）InnoDB锁

本小节介绍InnoDB使用的各种锁类型。

共享锁和独占锁

6、InnoDB配置指南

本节详细介绍InnoDB相关的配置，包括：初始化、启动、各个功能及组件的配置等。

（1）InnoDB启动配置