# MySQL 词汇表

这一部分主要解释一些在数据库服务器领域常用的专业术语。理解了这些术语，有助于了解数据库管理系统的实现，以及阅读各种技术文档。内容主要来自MySQL官方手册的MySQL词汇表这一部分。其中很多事InnoDB相关的。

**ACID: ACID**

原子性（atomicity）. 一致性（consistency）. 隔离性（isolation）和持久性（durability）的首字母缩写。这些属性是一个数据库系统必须全部具备的特性，并且与事务（***transaction***）的概念紧紧绑在一起。InnoDB的事务特性遵守ACID原则。

事务是可以提交或回滚的原子（***atomic***）工作单元。当一个事务造成数据库的多处更改时，所有的更改要么在事务提交（***committed***）后全部成功，要么在事务回滚（***rolled back***）后全部撤消。

数据库在任何时候都处于一致的状态——在每一次提交或回滚后，以及事务在进行时，如果相关的数据正在被跨表更新，这时select看到的是所有的旧值或所有的新值，而不会是新旧兼有的。

当多个事务在进行时，它们之间是被保护（被隔离）的；它们之间不能互相干涉或看到其它事务未提交的数据。这种隔离是靠锁（***locking***）机制实现的。有经验的用户可以调整隔离级别（***isoloation level***），当可以确认多个事务真的不会相互干涉时，你可以牺牲一定的隔离性以换取性能和并发（***concurrency***）能力的提升。

事务的结果是持久的：一旦提交成功了，那么在掉电、系统崩溃、资源竞争等危险情况下，事务引起的更改也不会丢失，数据库任然处于一致性的状态。持久性需要写到磁盘存储上。（在InnodDB中，双写缓冲（***doublewrite buffer***）来帮助完成一致性。）

**adaptive flushing: 自适应刷新**

一个InnoDB用来平摊由检查点（***checkpoints***）技术产年的I/O开销的算法。MySQL周期性地将一小部分修改了的页面（***pages***）刷新（***flushing***）到数据文件（***data files***）中，而不是一次性将所有修改过的页面从缓冲池（***buffer pool***）中刷新到数据文件。自适应刷新算法通过刷新率和redo信息产生的速率，估计出周期性刷新的最佳值。

**adaptive hash index: 自适应哈希索引**

是一个通过在内存中构建哈希索引（***hash index***）来加速对InnoDB表的=和IN查找操作的优化。MySQL监视InnoDB表的索引查询，如果查询更适合哈希索引，它会为频繁访问到的索引页（***pages***）自动创建一个哈希索引。从某种意义上讲，自适应哈希索引实时地配置MySQL，以充分利用内存空间，这样做更接近于内存数据库的架构。这个特性由innodb\_adaptive\_hash\_index这个选项控制。因为这个功能只在特定的负载（频繁访问某些热点数据）下受益，并且用于哈希索引的内存也保留在了buff pool中，所以你对这个功能应该权衡取舍。

哈希索引的创建基于InnoDB的已存在的以B树（***B-tree***）结构组织的二级索引（***secondary index***）。根据使用索引搜索模式的不同，MySQL可以在任何长度的B树键值前缀上构建哈希索引。一个哈希索引可以是部分的；整个B树没必要全缓存在buffer pool中。

在MySQL 5.6或更高的版本中，使用InnoDB表进行快速单值查找的另一种方法是使用InnoDB的 ***memcached***插件，具体如何使用可以参考MySQL的官方手册。

**Antelope**

原始InnoDB文件格式的code name。它支持冗余（***redundant***）与精简（***compact***）的行格式，但是不支持新的Barracuda中的动态（***dynamic***）与压缩（***compressed***）的行格式。

如果你的应用能从InnoDB表压缩功能中受益，或使用BLOBs或大文本字段时能从动态行格式中获益，你可以将表切到Barracuda格式。你可以通过在创建表之前设置innodb\_file\_format选项来选择使用何种文件格式。

**apply: 应用**

当一个MySQL企业版备份（MySQL Enterprise Backup）在数据库运行时生成的备份并没有包含最新的修改时，更新备份文件以包含这些改动的过程程就被称为应用（***apply***）步骤。它由mysqlbackup命令的apply-log选项指定。

在改动被应用前，我们把文件称为原始备份（***raw backup***）。在改动被应用后，我们把文件称为一致备份（***prepared backup***）。改动记录在\***ibbackup\_logfile**文件中；一旦应用步骤完成，这些文件就没啥用了。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [ibbackup\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibbackup_logfile), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [prepared backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_prepared_backup), [raw backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_raw_backup).

**asynchronous I/O: 异步I/O，也可说AIO**

一种允许其它操作可以在I/O完成之前继续的I/O操作类型，也被称为非阻塞I/O。简写为AIO。InnoDB在某些可以并行操作但不影响数据库准确性的操作上使用这种I/O类型，比如：把那些实际上没有访问到但马上就要被访问到的页读进buffer pool中。MySQL5.5开始在Linux上开始使用AIO了，因此依赖于libaio库。

**atomic: 原子（性）**

**atomic instruction: CPU中不可中断的指令（原子指令）**

CPU提供的特殊机制，用来保证关键的低层操作不被终端。

**auto-increment: 自增**

一个在列上自动添加递增序列值的表的列属性（用AUTO\_INCREMENT关键字指定）。InnoDB只支持主键列上的自增。

它会节省开发者的工作量，在插入新行时不必再生成新的唯一值。它为查询分析器提供很有用的信息，因为列已知是非空而且值唯一。这样的列可以在各种环境中用做查询的键，并且因为它们是自动生成的，所以根本没有必要改动它们；正因如此，主键列常常被定义为自增。

自增会对基于语句的复制（***statement-based replicataion***）带来麻烦， 因为在从库上同步过去的语句不一定会生成与主库上一致的值，这要归咎于计时问题。当你有一个自增主键，你只能在innodb\_autoinc\_lock\_mode=1的情况下使用基于语句的复制。如果你设置了innodb\_autoinc\_lock\_mode=2，因为它会允许高并发的插入操作，所以请使用基于行的复制（***row-based replication***）而不要使用基于语句的复制。innodb\_autoinc\_lock\_mode=0是之前的（传统的）默认设置，但只有遇到兼容性问题时才会使用该设置。

**auto-increment locking: 自增锁**

自增（auto-increment）主键在带来方便的同时会引起对并发性能上的损失。例如：在最简单的情况，如果一个事务正在往表里插值，其它的事务必须等待，为的是让第一个事务插入的行能够获得连续的键值。为此InnoDB包含一些优化以及innodb\_autoinc\_lock\_mode选项，这样你就可以选择如何在可预测的自增序列值与最大并发（concurrency）插入之间做出取舍。

**autocommit: 自动提交**

控制每句SQL语句后是否产生提交（commit）操作的选项。在InnoDB表中使用跨多条SQL语句的事务（transactions）的情况下，不推荐这个模式。它会帮助提高InnoDB表只读的事务（read-only transactions），这儿会将锁（locking）的负载和产生的undo数据最小化，特别是在MySQL 5.6.4及更高版本中。它同样适用于事务不起作用的MyISAM表。

**availability 可用性**

对MySQL、操作系统或硬件故障以及因为维护行为而引起宕机等主机故障的处理能力的描述，以及在必要情况下从上述所有故障中恢复的能力描述。经常与扩展性（scalability）配合使用，成为大规模部署中的一个关键因素。经常谈的高可用就是这个意思。

**B-tree: B树**

数据库索引上很流行的数据结构。该数据结构一直保持排序状态，保证了快速精确查找（等于操作）和范围查找（比如大于、小于和BETWEEN操作）。这种类型的索引在绝大多数的存储引擎中是可用的，如InnoDB和MyISAM。B树的节点可以有很多子节点，因此树的高度很低，这就可以减少磁盘的I/O次数。

哈希索引（***hash index***）与它的差别是，MySQL中哈希索引只在[MEMORY](https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html) 存储引擎中可用。[MEMORY](https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html) 存储引擎也可以使用B树索引，如果查询中用到了范围查找操作，你可以为[MEMORY](https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html) 表选用B树索引。

**backticks: 反引号**

MySQL SQL语句中的标识如果含有特殊字符或保留词，就必须用反引号（`）括起来。例如，为了使用名为SELECT列，你就要指定为`SELECT`。由于反引号提供一种额外的安全级别，它们被SQL语句广泛使用。其它的数据库系统使用双引号（"）将这样的特殊名字包围起来。为了移植性起见，你可以在MySQL启用ANSI\_QUOTES模式并用双引号来代替反引号来限定标识符名称。

**backup: 备份**

为了安全起见，拷贝MySQL实例部分或全部表的数据和元数据的过程。也指拷备完成的文件集合。这是DBA们的一项终极任务。这个过程的反向操作是恢复（***restore***）。

对于MySQL，物理备份（***physical backup***）由MySQL企业版备份（***MySQL Enterprise Backup***）产品来完成，逻辑备份（***logical backup***）由mysqldump工具来完成。这些技术所产生的备份数据在文件大小、文件结构以及速度（特别是恢复操作的速度）等方面都有不同的特性。

热备（***hot backup***）. 温备（***warm backup***）和冷备（cold backup）因它们干涉数据库操作的多少而有很大不同。（热备干涉最少，冷备最多。）

相关信息： [cold backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_cold_backup), [hot backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_hot_backup), [logical backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_logical_backup), [MySQL Enterprise Backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_mysql_enterprise_backup), [mysqldump](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_mysqldump), [physical backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_physical_backup), [warm backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_warm_backup).

**Barracda: Innodb code name**

支持表压缩的InnoDB文件格式的code name。这种文件格式首先是在InnoDB Plugin中提到的。它提供压缩行格式实现InnoDB的表压缩，提供动态行格式来提高BLOB和大文本字段的存储分布。你可以通过innodb\_file\_format选项来使用。

因为InnoDB系统表空间是以原始的Antelope文件格式存储的，所以要使用Barracuda文件格式时，你必须要启用file-per-table选项，它会将最新创建表的表空间从系统表空间中隔离出来。MySQL企业版备份产品的3.5版及以上版本支持用Barracuda文件格式来备份表空间。

更多信息： [Antelope](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_antelope), [compact row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_compact_row_format), [compressed row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_compressed_row_format), [dynamic row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_dynamic_row_format), [file format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_format), [file-per-table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_per_table), [general tablespace](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_general_tablespace), [innodb\_file\_format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_innodb_file_format), [MySQL Enterprise Backup](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_mysql_enterprise_backup), [row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_row_format), [system tablespace](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_system_tablespace).

**beta: 公测**

**binary log: 二进制日志**

包含所有尝试修改表数据语句记录的文件。你可以将二进制日志的功能打开或关闭，虽然Oracle建议你在用复制或备份时一直打开它。你可以通过mysqlbinlog命令来检查二进制日志的内容，或将这些语句在复制或恢复时重放。对于MySQL企业备份（***MySQL Enterprise Backup***）产品，二进制日志的文件名和文件中的当前位置是非常重要的细节。你可以用--slave-info选项来让复制环境下的备份文件记录主库的信息。

**binlog: 二进制日志（同binary log）**

二进制日志（***binary log***）的简写。例如，你会在邮件消息或论坛中看到这个简写。

**blind query expansion:**

由WITH QUERY EXPANSION子句启用的特殊模式的全文搜索。

**bottleneck: 瓶颈**

**bounce: 性能抖动**

关闭之后紧跟着重启。理想情况是提前做一个预热，这样性能和吞吐量会很快恢复到一个较高水平。

**buddy allocator （Innodb buffer pool内存分配中的单位）**

InnoDB的buffer pool管理不同大小的page的机制。

**buffer pool: 缓冲池**

缓存了InnoDB的表和索引数据的内存区域。为了提高大容量内存的读操作的效率，buffer pool以页面（***page***）为单位来管理内存，页面（***page***）是InnoDB中可以存储多行数据的内存块。为了提高缓存管理的效率，buffer pool将这些页面组织为一个链表；很少使用的数据利用LRU算法的变体从链表中老化并从缓存中剔出。在大内存的系统中，你可以通过将buffer pool分割为多个buffer pool实例（***buffer pool instance***）来提高并发度。

好几种InnoDB的状态变量。information\_schema表和performance\_schema表都能帮助你监测buffer pool的内部工作情况。从MySQL 5.6开始，你可以通过在服务器关闭时保存缓冲池的状态并在服务器启动时恢复到相同的状态，从而避免重新启动服务器之后较长时间的预热时间，对于具有大缓冲池的MySQL实例而言，这个效果更加明显。

**buffer pool instance: 缓冲池实例**

在buffer pool中划分的任何区域，由innodb\_buffer\_pool\_instances配置选项来控制。由innodb\_buffer\_pool\_size指定的总内存大小由所有的缓冲池实例瓜分。通常情况下，系统分配给InnoDB buffer pool多个GB情况下，才适合用多buffer pool instances（每个实例1GB或更大）。多个buffer pool instances可以提高数据库服务器的并发程度。

**built-in: 内置**

MySQL中内置的InnoDB存储引擎是存储引擎的原始分发形态。相对于InnoDB Plugin而言。自MySQL 5.5开始，InnoDB Plugin又合并回MySQL代码中成为内置的InnoDB存储引擎（也就是InnoDB 1.1）。

**cardinality: 基数**

表的列中不同值的数目。当一个查询引用到一些有索引（***index***）的列时，每个列的基数会影响到哪种访问方式是最有效的。例如，对于一个有唯一约束（***unique constraint***）的列，不同值的数目就等于表里的行数。如果一个表里有一百万行数据但某列只有10个不同的值，每一个值会（平均）出现十万次。像SELECT c1 FROM t1 WHERE c1 = 50;这样的查询，有可能返回一行，也有可能返回很多行，数据库服务器也可能会根据c1的**基数**来有区别地执行这个查询。如果列中的值分布是不均匀的，依靠**基数**来决择最优执行计划并不是个好方法。例如，SELECT c1 FROM t1 WHERE c1 = x;有可能当x=50时返回一行，而在x=30时返回一百万行。在这种情况下，你可能需要使用索引提示（***index hint***）来传递哪种方式对此类特定的查询更有效一些。基数也适用于多个列中的不同值的数量，比如在组合索引中。

更多信息：[column](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_column), [composite index](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_composite_index), [index](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_index), [index hint](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_index_hint), [persistent statistics](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_persistent_statistics), [random dive](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_random_dive), [selectivity](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_selectivity), [unique constraint](https://dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_unique_constraint).

**.cfg file: .cfg文件**

InnoDB的可传输表空间（***transportable tablespace***）特性所用到的元数据文件。它是由FLUSH TABLES ... FOR EXPORT命令产生的，这条语句将一个或多个表设置到一个统一的状态，这样它们可以被拷贝到其它服务器。.cfg文件会随着.ibd文件一起拷贝，并且在ALTER TABLE ... IMPORT TABLESPACE时被用于调整.ibd文件内部的值，比如 ***space ID***。

**change buffer：变更缓冲区**

一个记录二级索引（***secondary indexes***）中页（***pages***）上变化的特殊的数据结构。这些值可能是用SQL中的INSERT. UPDATE或DELETE语句（***DML***）引起的。这些特性加上变更缓冲区叫做变更缓冲，由插入缓冲（***insert buffer***）. 删除缓冲（***delete buffer***）和清除缓冲（***purge buffer***）组成。

当不在buffer pool中的二级索引中的相关页发生时，变更只被记录在变更缓冲区中。当相关的索引页被加载到buffer pool中且关联的变更还在变更缓冲区中时，这些page的变更会利用变更缓冲区中的数据应用到buffer pool（***merged***）中。在系统几近空闲或在缓慢关机时，清除（***purge***）操作会周期性地执行，将新的索引页写到硬盘上。清除操作可以将一个序列的索引值一起写到硬盘块中，这样做比将每一个值立即写到硬盘上要更有效。

从物理角度看，变更缓冲区是系统表空间（***system tablespace***）中的一部分，所以在数据库重启过程中索引的变更是被保持缓冲的。变量只有在页面因为其它读操作而被加载到buffer pool中时才会被应用（***merged***）。

存储在变更缓冲区中的数据的数量与类型由[innodb\_change\_buffering](https://github.com/xufubo/manual56/blob/master/docs/Chpater_14/14.02.06_InnoDB_Startup_Options_and_System_Variables.md##sysvar_innodb_change_buffering)和[innodb\_change\_buffer\_max\_size](https://github.com/xufubo/manual56/blob/master/docs/Chapter_14/14.02.14_InnoDB_Startup_Options_and_System_Variables.md#sysvar_innodb_change_buffer_max_size)配置选项来决定。要观察关于当前变更缓冲区中的数据，可以执行[SHOW ENGINE INNODB STATUS](https://github.com/xufubo/manual56/blob/master/docs/Chapter_15/13.07.05_SHOW_Syntax.md#13.07.05.16)命令。

以前叫插入变更区（***insert buffer***）。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [change buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffering), [delete buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_delete_buffering), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [insert buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffer), [insert buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffering), [merge], [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge), [purge buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge_buffering), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace).

**change buffering: 变更缓冲**

有关变更缓冲区（***change buffer***）功能的总称，由插入缓冲（***insert buffering***. 删除缓冲（***delete buffering***）和清除缓冲（***purge buffering***）构成。索引变更是由SQL语句引起，通常会带来随机I/O操作，变更会被“憋住”并用后台线程（***thread***）来周期性地执行。这种顺序操作可以将一个序列的索引值一起写到硬盘块中，这样做比将每一个值立即写到硬盘上要更有效。由[innodb\_change\_buffering](https://github.com/xufubo/manual56/blob/master/docs/Chpater_14/14.02.06_InnoDB_Startup_Options_and_System_Variables.md##sysvar_innodb_change_buffering)和[innodb\_change\_buffer\_max\_size](https://github.com/xufubo/manual56/blob/master/docs/Chapter_14/14.02.14_InnoDB_Startup_Options_and_System_Variables.md#sysvar_innodb_change_buffer_max_size)配置选项控制。

参见[change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [delete buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_delete_buffering), [insert buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffering), [purge buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge_buffering).

**checkpoint: 检查点**

当缓存在buffer pool中的数据页被变更后，为了减少磁盘IO，这些变更会在晚些时候写入到数据文件（***date files***）中，这个过程叫刷新（***flushing***）。检查点就是记录最后更改被成功写入到数据文件的位置（由LSN值来表示）。

**checksum: 校验**

InnoDB中一个用来检测表空间（***tablespace***）中的一个页（***page***）从磁盘读入到InnoDB buffer pool中时是否正确的验证机制。这个特性在MySQL5.5中可以通过innodb\_checksum配置选项来控制。innodb\_checksums在MySQL 5.6.3中被弃用，由innodb\_checksum\_algorithm替代。

innochecksum命令在MySQL服务关闭的情况下通过测试指定表空间（***tablespace***）文件的校验值来帮助诊断损坏的问题。MySQL也将校验和用于复制上。有关详细信息，请参阅配置选项binlog\_checksum，master\_verify\_checksum和slave\_sql\_verify\_checksum。

**child table: 子表**

在外键（***foreign key***）关系中，一个表中某一列的值引用了另一个表中对应列的值，前者就是子表。即：包含FOREIGN KEY ... REFERENCES子句以及可选的ON UPDATE和ON DELETE子句约束的表。父表中的相应行必须存在的情况下，才能在子表中创建对应的行。子表中的值可以阻止父表上的删除或更新操作，或者基于创建外键时使用的ON CASCADE选项，来引发子表中的自动删除或更新。

更多信息见：[foreign key](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_foreign_key), [parent table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_parent_table).

**clean page: 干净的页**

InnoDB buffer pool中，一个页面的所有变更都被写到（刷新到）数据文件（***date files***）时，这个页就叫干净的页。反之为脏页（dirty page）。

**clean shutdown**

一种关闭方式：没有任何错误，而且所有的改变在关闭之前都被应用到InnoDB表中，是慢关闭（***slow shutdown***）的同义词。与之相对的是崩溃或快速关闭（***fast shutdown***）。

**client: 客户端**

发送请求到服务器，并处理结果的一种程序类型。

**clustered index: 聚集索引**

InnoDB主键（***primary key***）索引术语。InnoDB表中数据行的存储是基于主键列中的值来组织的，用来加速涉及主键列的查询与排序。为了得到最优性能，请谨慎选择主键列。因为修改聚集索引是一个非常昂贵操作，请选择很少或从来不被修改的列为主键列（例如QQ号，微信号）。在Orcale的数据库产品中，这种表的类型被称为索引组织的表（***index-organized table***）。

**cold backup: 冷备份**

数据库在关闭状态下生成的备份（***backup***）。对于比较忙的应用和网站，这可能不太现实，你可以选择温备（***warm bakcup***）或热备（***hot bakcup***）。

**column: 列**

一行（***row***）中的一个数据项，它的存储和语义用数据类型来定义。每张表（***table***）和索引（***index***）主要由它所包含的列集来定义。每一列都有一个基数（***cardinality***）值。一个列可以是这个表的主键（***primary key***），也可以是主键的一部分。一个列可以受到唯一约束（***unique constraint***）. 非空约束（***NOT NULL constraint***），或两者共同约束。不同列中的值，即使在不同的表中，也可以用外键（***foreign key***）关系来联系起来。在讨论MySQL内部操作时，有时也用别名字段（***field***）来代替。

更多信息：[cardinality](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_cardinality), [foreign key](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_foreign_key), [index](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_index), [NOT NULL constraint](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_not_null_constraint), [primary key](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_primary_key), [row](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_row), [table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_table), [unique constraint](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_unique_constraint).

**column index 单列索引**

在单列上建的索引（***index***）。

**column prefix: 列的前缀**

当一个索引在创建时指定了长度，如CREATE INDEX idx ON t1 （c1（N）），只有这个列中值的前N个字符会被存储到索引中。保持小的索引前缀会使得索引精简，同时节省内存与磁盘I/O，有助于提高性能。（不过索引前缀太小的话，会妨碍查询优化，因为实际不同的值在前缀上却是相同的）。

对于含有二进制或大文本字符串的字段，排序不是主要的考虑因素，把全部的值都存到索引中也会浪费空间，索引会自动使用前N（一般来说是768）个字符来做查询与排序。

**commit: 提交**

用来结束一个事务（***transaction***）的SQL语句，使事务所做的更改持久化。与它相对的是回滚（***rollback***），它会将事务产生的任何更改撤销。InnoDB使用了乐观（***opimistic***）机制的提交，即：变更在提交操作真正发生之前可以写进数据文件中。这个技术让提交本身变得比较快，但代价是回滚操作需要花费更多的开销。这个假设我认为是合理的，实践中也确实是大部分事务是成功的，需要回滚的比较少。默认情况下，MySQL为自动提交，它可以在每一条SQL语句执行之后自动执行提交。

更多信息： [autocommit](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_autocommit), [optimistic](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_optimistic), [rollback](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_rollback), [SQL](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_sql), [transaction](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_transaction).

**compact row format: 紧凑（compact）的行格式**

MySQL 5.0.3之后5.7.8之前InnoDB默认的行格式（***row format***）。从MySQL 5.7.9起，默认行格式由innodb\_default\_row\_format配置选项定义。其默认设置为DYNAMIC。COMPACT行格式为空值和可变长度列提供比先前的默认值（REDUNDANT行格式）更紧凑的表示形式。该格式对于使用Antelope文件格式（***Antelope file format***）的表有效。有关InnoDB COMPACT行格式的其他信息，请参见[第14.11.4节“COMPACT和REDUNDANT行格式”](https://dev.mysql.com/doc/refman/5.7/en/innodb-row-format-antelope.html)，更多信息：

[Antelope](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_antelope), [dynamic row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_dynamic_row_format), [file format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_format), [redundant row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_redundant_row_format), [row format](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_row_format).

**composite index: 复合索引**

包含多列的索引。

**compressed backup: 压缩备份**

MySQL企业备份产品（***MySQL Enterprise Backup***）的一个压缩属性，它为每一个表空间生成一个压缩拷贝，并把后缀名从.ibd改为.ibz。压缩备份数据可以让你本机保留更多备份，并且节省将备份传输到其它服务上的时间。数据在恢复操作中解压。当压缩备份操作一个已经压缩过的表时，它会跳过对这个表的压缩步骤，因为再次压缩对空间的节省可能只是一小点或没有。

由MySQL企业备份产品（***MySQL Enterprise Backup***）生成的文件集，里面每个表空间（***tablespace***）都是压缩过的。压缩过的文件以.ibz后缀重命名。

备份从一开始就使用压缩有助于避开压缩过程中存储开销，同时可以避开往其它服务器上传输时的网络开销。应用（***apply***）二进制日志（***binary log***）的过程需要更多时间，并需要解压备份文件。

参见 [apply](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_apply), [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**compressed row format: 压缩的行格式**

一种可以让InnoDB表中数据和索引压缩的行格式（***row format***）。它做为\*\*\*Barracuda\*\*\*文件格式的一部分，在InnoDB Plugin中被引入。大字段从存储剩下字段的页中被隔离存储，如动态行格式（***dynamic row format***）。索引页和大字段都被压缩，节省了内存与磁盘。根据数据结构的不同，内存和磁盘使用量上的减少的好处是否能超过解压这些文件带来的性能开销还不好说。使用细节参见第[14.2.8节，InnoDB压缩表](https://github.com/xufubo/manual56/blob/master/docs/14.02.08)。

参见 [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [dynamic row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dynamic_row_format), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**compression: 压缩**

是一个使用更少磁盘空间. 执行更少I/O和使用更少用来缓存的内存的等好处多多的特性。InnoDB的表和索引数据可以在数据库操作过程中保持压缩格式。

当查询需要时数据被解压，当被\*\*\*DML\*\*\*操作改变时被重新压缩。当你对某个表启用了压缩了，这个过程对用户和应用开发者来说是透明的。DBA可以查阅\*\*\*information\_schema\*\*\*表来监控压缩的参数是如何为MySQL实例和特定的压缩表提供工作效率的。

当InnoDB表被压缩了，表（***table***）本身. 所有关联的索引（***index***）以及加载到\*\*\*buffer pool\*\*\*中的页都会被压缩。压缩不会被应用到\*\*\*undo buffer\*\*\*中的页中。

表压缩特性需要MySQL 5.5或更高版本，或InnoDB Plugin 5.1或更早版本，并且使用\*\*\*Barracuda\*\*\*文件格式和压缩行格式来创建表（\*\*\*innodb\_file\_per\_table\*\*\*选项要打开）。对每个表的压缩是受CREATE TABLE和ALTER TABLE语句的KEY\_BLOCK\_SIZE子句的影响。在MySQL 5.6及更高版本中，压缩也受服务范围的配置参数innodb\_compression\_failure\_threshold\_pct. innodb\_compression\_level和innodb\_compression\_pad\_pct\_max的影响。使用细节参考[第14.2.8节，InnoDB压缩表](https://github.com/xufubo/manual56/blob/master/docs/14.02.08)。

另一种压缩类型是压缩备份（***compression buckup***），是MySQL企业备份产品的特性。

参见 [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [INFORMATION\_SCHEMA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_information_schema), [innodb\_file\_per\_table], [plugin](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_plugin), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [undo buffer].

**compression failure 压缩失败**

不是一个真正的错误，更恰当的说法是在混合使用压缩（***compression***）和\*\*\*DML\*\*\*操作时产生的一个“昂贵”的操作。会在以下情况下发生：修改一个压缩过的页（***page***）溢出了为记录修改所预留的页；所有的变更都应用到表数据中，页面被再次压缩；重新压缩过的数据不再能适应原始的页，需要MySQL分割数据到两个新的页并对其单独分别压缩。为了检查这种情况发生的频率，查询表INFORMATION\_SCHEMA.INNODB\_CMP并检查有多少COMPRESS\_OPS列的值超过COMPRESS\_OPS\_OK列上的值。理想情况下，压缩失败并常经常出现；当它们出现时，你可以调整配置选项[innodb\_compression\_level]. [innodb\_compression\_failure\_threshold\_pct]和[innodb\_compression\_pad\_pct\_mac]。

参见 [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**concurrency: 并发**

多个操作（在数据库的术语中，叫\*\*\*transactions\*\*\*）同时执行而不会互联影响的能力。并发也与性能是密切相关的，因为在理想情况下，使用了高效的锁机制（***locking***）来以最小的性能代价去保护多个同时工作的事务。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [transactions].

**configuration file: 配置文件**

保存MySQL启动参数选项（***option***）的文件。传统上该文件在Linux/UNIX上名为my.cnf，在Windows上名为my.ini。你可以在该文件的mysqld节设置大量与InnoDB相关的选项。一般来讲，这个文件可以在/etc/my.cnf、/etc/mysql/my.cnf、 /usr/local/mysql/etc/my.cnf和~/.my.cnf下找到。有关该文件的搜索路径的细节请参见[第4.2.6节“使用选项文件”](https://dev.mysql.com/doc/refman/5.7/en/option-files.html)。

当你使用MySQL企业备份（***MySQL Enterprise Backup***）产品时，你一般会用到两个配置文件：一个用来指定数据从哪儿来工它们是如何组织的（它可能是你真正服务的最原始的配置文件），另一个是只包含一小部分选项的精简版，用来指定备份数据去哪儿和它们是如何组织的。要使用MySQL企业备份（***MySQL Enterprise Backup***）产品的话，配置文件中必须包含常规配置文件中一些没有加进去的选项，所以为使用MySQL企业备份（***MySQL Enterprise Backup***）产品起见，你可能要向已存在的配置文件中加入一些选项。

参见 [my.cnf](https://github.com/xufubo/manual56/blob/master/docs/mycnf), [option file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_option_file).

**consistend read: 一致性读**

一个使用快照信息来呈现基于一个时间点的查询结果的读操作，而不管同一时间点上执行的事务所带来的变更。如果查询到的数据已经被其它事务改变，原始数据会基于\*\*\*undo log\*\*\*中的内容进行重建。这个技术避免了一些因为一个事务被强制等待另一个事务结束而降低并发（***concurrency***）的加锁（***locking***）问题。

在可重复读隔离（***repeatable read***）级别，快照基于第一次读操作执行的时间。在提交可读（***read committed***）隔离级别，快照在每一次一致性读操作时都重置。

一致性读的默认模式下，InnoDB会用提交可读（***READ COMMITTED***）和可重复读（***REPEATABLE READ***）隔离级别处理SELECT语句。因为一个一致性读操作不会向它访问的表加任何锁，所以其它会话可以在一个一致性读操作正在这个表上执行的时候自由地修改这些表。

更多可用的隔离级别的技术细节，请参考[第14.2.2.4节，一致性无锁读](https://github.com/xufubo/manual56/blob/master/docs/14.02.02.04)。

参考 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [MVCC](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mvcc), [READ COMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_committed), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**constraint: 约束**

一个可以阻止数据库更改以防止数据变得不一致的自动测试。（在计算机科学术语中，是一种与不变状态相关的断言）。约束是\*\*\*ACID\*\*\*理念中至关重要的组成部，用来维持数据的一致性。MySQL中支持约束的有外键约束（***FOREIGN KEY constraints***）与唯一约束（***UNIQUE constraints***）。

参考 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key), [relational](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_relational), [unique constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_constraint).

**counter: 计数器**

由一种特殊的InnoDB操作增加的一个值。有助于标志一个服务繁忙程度. 分析性能问题的源头和测试变更（例如，对配置选项或查询使用的索引的变更）是否有期望的低级别的效果。不同类型的计数器可以通过\*\*\*performance\_schema\*\*\*表和\*\*\*infomation\_schema\*\*\*表，特别是infomation\_schema.innodb\_metrics表来获得。

参见 [INFORMATION\_SCHEMA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_information_schema), [metrics counter](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_metrics_counter), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema).

**covering index: 覆盖索引**

一个包含查询所检索的所有列的索引（***index***）。替代将索引值当成指针在全表行里查找，查询直接从索引结构中返回值，节省了磁盘I/O。InnoDB要比MyISAM能更多地应用这种优化技术，因为InnoDB的二级索引（***secondary index***）也包含了主键列。InnoDB不能在被一个事务修改过的表上应用这种技术，直到这个事务结束为止。

给定一下正确的查询，任何单列索引（***column index***）或组合索引（***composite index***）都可以做为一个覆盖索引。设计你的索引或查询让它们能够在任何可能的情况下利到这种优化技术的好处。

参见 [column index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_column_index), [composite index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_composite_index), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index).

**crash: 崩溃**

MySQL使用“崩溃”这个术语来一般指代服务的任何在没有做自己正常清理工作情况下的非预期的宕机（***shutdown***）操作。例如，一个崩溃可能由数据库服务器上机器或存储设备的硬件故障引发；一个潜在的数据不匹配导致MySQL服务的挂起；一个DBA触发的快速关机（***fast shutdown***）；或其它更多原因。\*\*\*InnoDB\*\*\*自动崩溃恢复（***crash recovery***）的鲁棒性确保在服务重起后数据是一致的，而不需要DBA做额外的工作。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [fast shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_shutdown), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shutdown).

**crash recovery 崩溃恢复**

MySQL在崩溃后再次启动时做的清理行为。对于InnoDB表，未完成的事务带来的变更会利用redo log中的数据来重放。在崩溃之前提交的但尚未写到数据文件中的变更会从双写缓冲中重构。当数据库正常关机时，此类行为会在关闭时由清除操作来执行。

在正常的操作中，提交了的数据在写入到数据文件前的一段时间内被存储在变更缓冲中。这在与一直让数据文件保持最新之间存在取舍，让数据保持最新会在正常操作中带来性能开销，但缓冲数据会在关机或崩溃恢复时要花费更多时间。

参见 [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [crash](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log).

看了

**CRUD**

“create, read, update, delete”首字母缩写，数据库应用中的常见操作序列。

**cursor: 游标**

一个用来表示查询（***query***）结果集的内部数据结构，或其它的使用SQL WHERE子句执行搜索的操作。它与高级语言中的迭代器一样，指向结果集中的每个值。通常情况下，不需要直接处理游标，只是在某些性能问题中，需要详细研究它。

**data dictionary: 数据字典**

保存诸如表（***tables***）、索引（***indexes***）以及表列（***columns***）等InnoDB相关对象的元数据。这些元数据的物理位置在InnoDB系统表空间（***system tablespace***）中。因为历史原因，它与存储在 .frm文件中的信息在某些维度上是重合的。因为MySQL企业备份（***MySQL Enterprise Backup***）产品一直会备份系统表空间，所以所有的备份都包含数据字典的内容。

**data directory: 数据目录**

一个目录，下面存储着每个MySQL实例（***instance***）保存InnoDB数据的文件（***data files***）和每个数据库对应的目录。该目录的地址由datadir配置选项控制。

**data files 数据文件**

在物理上包含InnoDB表（***table***）和索引（***index***）数据的文件。InnoDB系统表空间（***system tablespace***）包含InnoDB的数据字典，也可以包含多个InnoDB表的数据，由一个或者多个.ibdata文件表示，该文件直接位于数据目录中（由datadir指定的值）。并不位于某个数据库目录中。例如：

**MySQL [(none)]> show variables like '%datadir';**

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

| Variable\_name | Value |

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

| datadir | /var/lib/mysql/ |

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

**[xufubo@localhost mysql]$ pwd**

/var/lib/mysql

**[xufubo@localhost mysql]$ ls**

auto.cnf ib\_logfile0 performance\_schema

ca-key.pem ib\_logfile1 private\_key.pem

ca.pem ibtmp1 public\_key.pem

client-cert.pem localhost.localdomain.pid server-cert.pem

client-key.pem mysql server-key.pem

ib\_buffer\_pool mysql.sock sys

ibdata1 mysql.sock.lock test

[xufubo@localhost mysql]$

这时会有表示系统表空间的文件ibdata1，直接位于数据目录中。

创建一个test数据库，将会创建一个目录test。

当file-per-table选项激活时，新创建的表将被存储到单独的File-per-table 表空间中，即File-per-table表空间存放单独的InnoDB表，由ibd文件表示，而且该文件位于数据库目录中（因为此时一个表空间对应一个InnoDB表）。例如：

MySQL [test]> show variables like 'innodb\_file\_per\_table';

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

| Variable\_name | Value |

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

| innodb\_file\_per\_table | ON |

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

MySQL [test]> create table t1(c1 int);

这时test目录下有如下文件：

[xufubo@localhost test]$ sudo ls

db.opt t1.frm t1.ibd

其中db.opt存放数据库test的元数据（包括数据库的编码字符集），t1由MySQL服务器层创建，为表的元数据，t1.ibd用于存放该表的数据和索引。

一般表空间（在MySQL 5.7.6中引入）可以容纳多个InnoDB表的数据，也由.ibd数据文件表示。

更多信息：[data dictionary](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_data_dictionary), [file-per-table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_per_table), [general tablespace](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_general_tablespace), [.ibd file](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_ibd_file), [ibdata file](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_ibdata_file), [index](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_index), [system tablespace](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_system_tablespace), [table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_table), [tablespace](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_tablespace).

**data manipulation language: DML**

**data warehouse数据仓库**

**database: 数据库**

在MySQL数据目录（***data directory***）中，每个数据库对应一个单独的目录。InnoDB的系统表空间（***system tablespace***）可以存储MySQL实例（***instance***）中多个数据库的表的数据，被保存在多个数据库目录之外的数据文件中，例子见上边。当file-per-table模式激活时，默认情况下，代表每个InnoDB表的.ibd文件存储在对应的数据库目录中。除非使用DATA DIRECTORY子句指定目录。每个InnoDB表一个.ibd文件。一般表空间（General tablespaces）的数据也由.ibd文件表示，可以存储MySQL实例（***instance***）中多个数据库的表的数据，被保存在多个数据库目录之外的数据文件中。并且可以被保存在独立于MySQL数据目录的目录中。

对于长时间使用MySQL的用户来说，数据库是一个熟悉的概念。一个有Oracle数据库背景的用户会发现MySQL中数据库的意思与Oracle数据库中的schema很相近。

**DCL: Data control language**

数据控制语言，一组用来管理权限的SQL语句，由GRANT和REVOKE语句组成。与DDL和DML对应。

**DDL: Data definition language**

数据定义语言，一组用来操作数据库本身而不是单独表行的SQL语句。包括CREATE、ALTER和DROP语句。也包括TRUNCATE语句，尽管从最终效果上看与DELETE FROM tabel\_name非常相似，但是实现原理完全不同。DDL语句自动提交（***commit***）当前事务（***transaction***）；它们不能回滚（***rolled back***）。

InnoDB的在线DDL功能增强了CREATE INDEX，DROP INDEX和许多类型的ALTER TABLE操作的性能。有关详细信息，请参见[第14.13节“InnoDB和在线DDL”](https://dev.mysql.com/doc/refman/5.7/en/innodb-online-ddl.html)，此外，InnoDB的 [file-per-table](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_per_table) 设置可能会影响DROP TABLE和TRUNCATE TABLE操作的行为。

以下的还没看。

**deadlock: 死锁**

不同的事务不能够继续的一种情况，因为它们都持有一个其它事务（***transactions***）所需要的锁（***lock***）。由于两个事务都需要等待资源可用，也永远不会释放他们所持有的锁。

当一个事务以相反的顺序锁定了多个表中的行（由诸如UPDATE或SELECT .. FOR UPDATE）时会发生死锁。一个死锁也有可能在诸如锁定索引的范围和间隙时发生，每个事务获取了一些锁但因为时间原因没有获取到其它锁。

为了减少死锁的几率，使用事务，而不是LOCK TABLE语句；保持事务中插入或更新的数据足够小这样它们就不会过久地停留在打开状态；当不同的事务更新多个表或大范围的行时，在每个事务中使用相同的操作顺序（如SELECT ... FOR UPDATE）；为SELECT ... FOR UPDATE和UPDATE ... WHERE语句创建索引。死锁几率与隔离级别（***isolation level***）无关，因为隔离级别改变读操作行为，而死锁因写操作而起。

如果一个死锁产生了，InnoDB检测到状态并让其中一个事务（***victim***）回滚（\***rolls back**）。因此，就算是你的应用逻辑准确无比，你也应该处理一个事务需要重试的情况。要查看InnoDB用户事务中的最后一个死锁，使用SHOW ENGINE INNODB STATUS。如果频繁的死锁让一个事务结构体或应用错误处理的问题变得非常明显，激活innodb\_print\_all\_deadlocks选项运行mysqld，它会将所有关于死锁的信息全打到mysqld错误日志中。

更多死锁如何自动检测与处理的背景资料，参考[第14.2.2.10节，死锁检测与回滚](https://github.com/xufubo/manual56/blob/master/docs/Chpater_14/14.02.02_InnoDB_Concepts_and_Architecture.md#14.02.02.10)。更多避免与恢复死锁状况的提示，参考[第14.2.2.11节，如何应对死锁](https://github.com/xufubo/manual56/blob/master/docs/Chpater_14/14.02.02_InnoDB_Concepts_and_Architecture.md#14.02.02.11)。

参考 [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [gap](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [victim](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_victim).

**deadlock detection: 死锁检测**

一个自动检测死锁（***deadlock***）并自动回滚（***roll back***）其中一个事务（***the victim***）的机制。可以使用innodb\_deadlock\_detect配置选项禁用死锁检测

**delete: 删除**

当InnoDB处理删除语句时，行记录被标记为删除，并不再被查询结果返回。真正的删除操作由单独的线程定期的执行，释放存储空间。

**delete buffering 删除缓冲技术。**

将因删除操作而发生的索引变更在插入缓冲（***insert buffer***）中存储而不是立马写入它们的技术，这么做是为了将物理写操作的随机I/O降到最小。（因为删除操作有两步处理，这个操作将正常标记索引记录为删的写操作缓冲下来。）它是变更缓冲（***change buffering***）的一种类型；其它的为插入缓冲（***insert buffering***）和清除缓冲（***purge buffering***）。

参见 [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [change buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffering), [insert buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffer), [insert buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffering), [purge buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge_buffering).

**denormalized: 反范式**

一种数据存储策略，它将多个不同表中的数据重复复制，而不是将这些表通过外键（***foreign keys***）和关联（***join***）查询链接起来。一般在数据仓库（***data warehouse***）应用中使用，它的数据在加载后不会再更新。在这样的应用中，查询的性能要比为更新时维持一致更重要而使其简单更重要。

与它对应的是范式（***normalized***）。

参见 [data warehouse](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_warehouse), [normalized](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_normalized).

**descending index: 降序索引**

在某些数据库系统中有效的一种索引，索引的存储专门为处理ORDER BY column DESC子句而优化。虽然目前MySQL允许DESC关键字出现在CREATE TABLE语句中，但它并没有为结果索引使用用任何特殊的存储布局。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index).

**dirty page: 脏页**

一个已经在内存中被更新过，但变更尚未写入（刷新,***flushed***）到数据文件（***data files***）中的InnoDB \*\*\*buffer pool\*\*\*中页（***page***）。

与它对应的是净页（***clean page***）。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [clean page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clean_page), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**dirty read: 脏读**

一个获取到不可靠数据的操作，数据被另一个事务更新但尚未提交。它只在未提交读的隔离级别中有可能出现。

这种操作并不遵守数据库设计的ACID原则。它的一致性是非常有风险的，因为数据可能会被回滚，或在提交前进一步被更新；那个时候，做了脏读的事务可能会使用从未被准确确认过的数据。

它的对立面是一致性读，InnoDB全力确保一个事务不会读到另一个事务修改过的信息，即使其它事务在期间已经做了提交。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [READ COMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_committed), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback).

**disk-based 基于磁盘**

See Also adaptive hash index, buffer pool, in-memory database. 主要在磁盘（硬盘或等同于硬盘）上组织数据的一种数据库。数据在磁盘与内存之间来回传输修改。与它对应的是内存数据库。尽管InnoDB是基于磁盘的，但经也包含一些诸如buffer pool. 多buffer pool实例和自适应索引等让一些工作主要在内存中完成。

参见 [adaptive hash index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_adaptive_hash_index), [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [in-memory database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_in_memory_database).

**cpu-bound: CPU受限**

一种瓶颈（***bottleneck***）主要是内存中CPU操作的负载类型。通常来说会包括读密集型的操作，其中的结果可以全部缓存中\*\*\*buffer pool\*\*\*中。

参见 [boteneck], [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [disk-bound](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_disk_bound), [workload](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_workload).

**disk-bound: 磁盘受限**

一种瓶颈（***bottleneck***）主要是磁盘I/O的负载类型。（也叫I/O带宽,***I/O-bound***。）一般包括频繁写盘或随机读取更多不适合放在\*\*\*buffer pool\*\*\*中的数据。

参见 [bottleneck](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_bottleneck), [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [cpu-bound], [workload](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_workload).

**DML: 不译**

数据操作语言，一组用来执行insert. update和delete操作的SQL语句。SELECT语句有时候也被当成是DML句句，因为SELECT ... FOR UPDAET模式受到与INSERT. UPDATE和DELETE一样的锁的考虑。

DML语句对InnoDB表的操作是以事务方式进行的，所以它的效果可以当成一个单元被提交或回滚。

对比使用\*\*\*DDL\*\*\*和\*\*\*DCL\*\*\*。

参见 [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [DCL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dcl), [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback), [SQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sql), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction)。

**document id: 全文索引编号**

在InnoDB全文搜索（***full-text search***）特性中，一个在表中包含全文索引（***FULLTEXT index***）的特殊的列，用来唯一标识与每个搜索词链表（***ilist***）相关联的文档。它的名字为FTS\_DOC\_ID（必须大写）。此列本身类型必须为BIGINT UNSIGNED NOT NULL，并有一个名为FTS\_DOC\_ID\_INDEX的唯一索引。你最好在创建表的时候就定义此列。如果InnoDB必须加入此列，同时创建一个全文（***FULLTEXT***）索引，索引操作是相当昂贵的。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index), [ilist](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ilist).

**doublewrite buffer: 双写缓冲**

InnoDB使用一种新的文件刷新技术叫双写。在页（***pages***）写入到数据文件（***data files***）之前，InnoDB先把它们写到相邻的一个叫做双写缓冲的区域中。只有当写入并刷新到双写缓冲的操作已经完成的情况下，InnoDB才会将这些页写到数据文件中它们真正的位置上。如果操作系统在页写的中途崩溃了，InnoDB可以在稍后的崩溃恢复（***crash recovery***）中从双写缓冲找到一个健全的页的拷贝来。

尽管数据始终要写两次，但双写缓冲并不需要两部的I/O开销或I/O操作。数据自身被当成一个大的连续的块写入到缓冲中，并用一个单独的fsync（）调用操作系统。

要关掉双写缓冲，设置选项innodb\_doublrwrite=0。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge).

**drop: 删除**

一种通过诸如DROP TABLE或DROP INDEX等语句来删除数据库对象的\*\*\*DDL\*\*\*操作。内部映射为ALTER TABLE语句。从InnoDB角度看，这些操作的性能考虑主要包括为确保相关的对象全部被更新而锁定数据字典（***data dictionary***）的时间，以及更新如\*\*\*buffer pool\*\*\*等内存结构的时间。对于一个表（***table***）来说，drop操作跟\*\*\*truncate\*\*\*操作（TRANCATE TABLE语句）在特性上有一点儿不一样。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [truncate](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_truncate).

**dynamic row format: 变长行格式**

InnoDB Plugin中引入的一种行格式，是Barracuda文件格式（***Barracuda file format***）中的一部分。因为TEXT和BLOB字段在保持其它字段的页之外存储，所以它对包含大对量的行来说非常高效。因为大字段通常不会被访问用来做查询条件，所以他们经常不会被带入到buffer pool中，从而减少I/O操作，并更合理地利用了缓存。

参见 [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [file format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_format), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**E**

**early adopter: 测试版**

类似测试（***beta***），是一个产品在非关键任务环境中通过了性能. 功能和兼容性评估的阶段。InnoDB用\*\*\*early adopter\*\*\*来代替beta，通过一系列的更新版本，达到\*\*\*GA\*\*\*发布版本。

参见 [beta](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_beta), [GA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ga).

**error log: 错误日志**

一种展示MySQL启动. 关键运行时错误及崩溃（***crash***）信息的日志（***log***），细节请参考[5.2.2节，错误日志](https://github.com/xufubo/manual56/blob/master/docs/05.02.02)。

参见 [crash](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash), [log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log).

**eviction: 替换出去**

一个将项目从缓存或其它临时存储区删除的过程，比如从InnoDB \*\*\*buffer pool\*\*\*中。通常但不总时用\*\*\*LRU\*\*\*算法来判断将哪个项目删除。当一个脏页（***dirty page***）被替换出去，它的内容会被刷新（***flushed***）到磁盘，并且任何脏（***dirty***）的相邻（***neighbor***）页也可能会被刷新。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [LRU](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lru).

**exclusive lock: 排它锁**

一种阻止任何其它的事务（***transaction***）锁定相同行的锁（***lock***）。取决于事务隔离级别（***isolation level***），这种类型的锁可能阻止其它事务写相同的行，也有可能阻止其它事务读取相册的行。InnoDB默认的隔离级别，可重复读（***REPEATABLE READ***），通过允许事务读取加有排它锁的行获得更高的并发（***concurrency***），这种技术叫做一致性读（***consistent read***）。

参见 [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [shared lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shared_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**extent: 簇**

表空间（***tablespace***）中总共1MB的一组页（***pages***）。默认的页大小（***page size***）为16KB，一个区包含64个页。在MySQL5.6中，页大小可以是4KB或8KB，这种情况下一个簇可以包含更多的页，但总大小仍为1M。

诸如段（***segments***）. 预读（***read-ahead***）请求和双写缓冲（***doublewrite buffer***）等这样InnoDB的特性在使用读. 写. 申请或释放数据时，都是一次一个簇来操作。

参见 [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer), [neighbor page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_neighbor_page), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [page size](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page_size), [read-ahead](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_ahead), [segment], [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**F**

**Fast Index Creation: 快速索引创建**

一种通过避免完全重写对应的表而加速InnoDB二级索引（***secondary indexes***）创建速度的功能，在InnoDB Plugin中引入，一在是MySQL 5.5及更高版本的组成之一。加速同样适用于删除二级索引。

因为索引的维护会增加性能上的开销来完成大量的数据传输操作，所以可以考虑在做诸如ALTER TABLE ... ENGINE=INNODB或INSERT INTO ... SELECT FROM ...的操作时不做二级索引，而在完成后再创建索引。

在MySQL 5.6中，这个特性变得更加通用：你可以在索引正在被创建时读写这个表，并且更多类型ALTER TABLE操作可以在不拷贝表. 不阻止\*\*\*DML\*\*\*操作或两者兼有的情况下执行。因此我们可以将这组特性叫做在线DDL而不是快速索引创建。

参见 [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [online DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online_ddl), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index).

**fast shutdown: 快速关闭**

InnoDB默认的关闭（***shutdown***）程序，基于配置设置innodb\_fast\_shutdown=1。为了节省时间，某些刷新（***flush***）操作被跳过。这类关闭在正常使用中是安全的，因为刷新操作会在下次启动时执行，使用了与崩溃恢复（***crash recovery***）相同的机制。在因为升级或降级而关闭数据库的情况下，做缓慢关机（***slow shutdown***）替代以确保关机过程中所有相关的变更都应用到数据文件（***data files***）中。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shutdown), [slow shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slow_shutdown).

**file format: 文件格式**

InnoDB为每个表所使用的格式，通常激活\*\*\*file-per-table\*\*\*设置来确保每张表都存储在单独的.ibd文件（\*\*\*.ibd file\*\*\*）中。目前，InnoDB中可用的格式是\*\*\*Antelope\*\*\*和\*\*\*Barracuda\*\*\*。每种文件格式都支持一种或多种行格式（***row format***）。Barracuda表中可用的行格式，压缩（***COMPRESSED***）与动态（***DYNAMIC***），为InnoDB开启了重要的新的存储特性。

参见 [Antelope](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_antelope), [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [ibdata file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibdata_file), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**file-per-table: 独立表空间**

受innodb\_file\_per\_table选项控制的设置的普通名。这是一个影响InnoDB文件存储. 功能可用性及I/O好多方面的重要配置。在MySQL 5.6.7及更高版本中它默认开启。在MySQL 5.6.7之前的版本中它默认关闭。

对于每一个在此选项生效时创建的表来说，数据是存储在一个单独的.ibd文件（\*\*\*.ibd file\*\*\*）中，而不是系统表空间（***system tablespace***）的ibdata文件（***ibdata file***）中。当表数据存储在独立的文件中时，你有更多选择非默认文件格式（***file format***）与行格式（***row format***）的灵活性，这些格式要求一些必须的特性，如数据压缩。TRUNCATE TABLE操作也会更快，并且回收了的空间可以为操作系统所用，而不是留给InnoDB。

MySQL企业备份产品（***MySQL Enterprise Backup***）对它们自己文件中的表来说更具灵活性。比如，表在单独的文件中时，它可以从备份中排除掉。所以，这个选项适合那些备份频率降低或在不同的备份日程中的表。

参见 [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [file format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_format), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [ibdata file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibdata_file), [innodb\_file\_per\_table], [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace).

**fill factor: 填充因子**

在InnoDB索引（***index***）中，在页（***page***）分裂前页中被索引数据占据的部分。在索引数据最初在页间划分时没有用到的空间允许那些用更长的字符串更新的行不再需要代价过高的索引维护操作。如果填充因子过小，索引需要消耗比所需更多的空间，导致在读索引时带来额外的I/O开销。如果填充因子过高，任何增加列值长度的更新都会导致因索引维护带来的额外的I/O负载。更多信息参考[第14.2.2.13.4节，InnoDB索引的物理结构]。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**fixed row format: 定长行格式**

MyISAM引擎使用的行格式，InnoDB不用。如果你使用row\_format=fixed选项创建一张InnoDB表，InnoDB会把这个选项转化成精简行格式（***compact row format***）来替代，不过fixed变量可能仍然出现在诸如SHOW TABLE STATUS报表中。

参见 [compact row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compact_row_format), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**flush: 刷新**

将变更写到数据库磁盘文件中，这些变量是已经缓冲在一个内存区或一个临时磁盘存储区。被定期刷新的InnoDB的存储结构体包括\*\*\*redo log\*\*\*. \*\*\*undo log\*\*\*和\*\*\*buffer pool\*\*\*。

刷新会因为内存变满了和系统需要释放一些空间而发生，因为一个提交（***commit***）操作意味着来自事务的变更可以确定了，或因为一个缓慢关机（***slow shutdown***）操作意味着所有的未完成的工作应该确定了。在并不需要一次性将所有缓冲了的数据刷新时，InnoDB使用一种叫模糊检查点（***fuzzy checkpointing***）的技术刷新小批量的页来平滑I/O负载。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [fuzzy checkpointing](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fuzzy_checkpointing), [neighbor page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_neighbor_page), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [slow shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slow_shutdown), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**flush list: 刷新列表**

一个InnoDB内部的用来跟踪\*\*\*buffer pool\*\*\*中脏页（***dirty page***）数据结构体：就是已经被变更的并且需要被写回到磁盘的页（***pages***）。这个数据结构体被InnoDB内部的迷你事务（***mini-transactions***）频繁更新，所以被自身的互斥锁（***mutex***）保护来允许对buffer pool的并发访问。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [LRU](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lru), [mini-transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mini_transaction), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [page cleaner](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page_cleaner).

**foreign key: 外键**

不同InnoDB表的行之间的一种指针关系的类型。外键关系是在父表（***parent table***）和子表（***child table***）的一个列上建立的。

除了能快速查找相关相信外，外键可以防止指针因插入. 更新和删除变得无效，有助于强制引用一致性（***referential integrity***）。这种强制是一种约束（***constraint***）。如果相关联的外键值在其它表里不存在的话，指向它的行将不能插入。如果一行被删除了或它的外键值被修改了，并且其它表中的有行指向这个外键值，外键可以被设置为阻止删除. 让其它表中相符的的列值变为空或自动删除其它表中相符的行。

设计一个范式化（***normalized***）数据库的阶段之一是找出重复的数据，将这些数据分离到一个新表中，设置一个外键关系，使用关系操作，这样就可以像查询一张表一样来查询多张表了。

参见 [child table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_child_table), [FOREIGN KEY constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key_constraint), [join](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_join), [normalized](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_normalized), [NULL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_null), [parent table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_parent_table), [referential integrity](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_referential_integrity), relational.

**FOREIGN KEY constraint: 外键约束**

通过一个外键关系（***foreign key***）来维护数据库一致性的约束类型。像其它约束一样，如果数据可能会变得不一致时，它可以阻止数据的插入与更新；在这种情况下，被阻止的不一致性是在多个表的数据之间的。或者说，当一个\*\*\*DML\*\*\*操作被执行了，基于在创建外键时指定的ON CASECADE选项，外键约束可以导致字表中的行（***child rows***）被删掉. 变更为不同的值或设置为空。

参见 [child table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_child_table), [constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_constraint), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key), [NULL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_null).

**.frm file: .frm file**

一个包含MySQL表的诸如表定义等元数据的文件。

对于备份来说，你得必须一直随着备份数据保存所有.frm文件的集合，这样可以恢复那些在备份后被修改或删掉的表。

尽管每张InnoDB表都有一个.frm文件，但InnoDB在系统表空间中维护了自己表的元数据；对于InnoDB来说操作InnoDB表不再需要.frm文件了。

这些文件会被MySQL企业备份产品（***MySQL Enterprise Backup***）备份。这些文件在正在备份时必须不被ALTER TABLE操作修改，这就是为什么包含非InnoDB表的备份要在备份.frm文件时执行一个FLUSH TABLES WITH READ LOCK操作来冻结这种活动的原因。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup).

**FTS: 全文搜索**

在大多数据情况下，是全文搜索（***full-text search***）的首字母缩写。有时在讨论性能问题时，也是全表扫描（***full table scan***）的缩写。

参见 [full table scan](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_table_scan), [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search).

**full backup: 全备**

包含一个MySQL实例（***instance***）中所有数据库，每个数据库（***database***）中所有表（***tables***）的备份（\***backup**）。相对的是部分备份（***partial backup***）。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [instance](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_instance), [partial backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_partial_backup), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**full table scan: 全表扫描**

一个没有使用索引选择部分数据，而是需要读取整表内容的操作。通常是在小的查询表中执行，或在数据仓库环境下的大表中执行，其中所有有效的数据都要被聚合和分析。这些操作出现的频繁程度，以及这些表大小所对对应的可用内存，对于查询优化与管理buffer pool都有影响。

索引（***indexes***）的目录就是为了允许在大表中查找特定的值或一定范围的值，这样就避免了全表扫描。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [LRU](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lru).

**full-text search: 全文搜索**

在表数据中查找单词. 词组及单词布尔组合等的MySQL特性，比你使用SQL LIKE操作符或在自己应用层写搜索算法的方式要更快. 更方便和更灵活的方式。它使用SQL的MATCH（）函数和全文索引（***FULLTEXT indexes***）。

参见 [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**FULLTEXT index: 全文索引**

保存MySQL全文索引机制中搜索索引的特殊的索引类型。包含了列中删除停用词之后的单词。最早只在MyISAM表中可用，自MySQL 5.6.4起，InnoDB表中也可以使用了。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [search index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_search_index), [stopword](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_stopword).

**fuzzy checkpointing 模糊检查点刷新**

一种从\*\*\*buffer pool\*\*\*中使用小批量刷新（***flush***）脏页（***dirty page***），而不是使用可能会打断数据库进程的一次性刷新所有脏页的技术。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush).

**G**

**GA: 一般可用（建议直接用 GA）**

“一般可用（Generally available）”，是软件产品离开测试版并且可供销售. 官方支持及生产可用的阶段。

参见 [beta](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_beta), [early adopter](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_early_adopter).

**gap: 间隙**

InnoDB索引（***index***）数据结构中新值插入的地方。当你使用诸如SELECT ... FOR UPDATE这样的语句锁定一个行集时，InnoDB会创建应用到间隙和真实值的锁。比如，如果你选定了大于10的所有值来更新，间隙锁会阻止其它事务插入大于10的新值。上确界记录（***supremum record***）与下确界记录（***infimum record***）代表了包含所有大于或小于所有当前索引值的间隙。

参见 [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [gap lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap_lock), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [infimum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_infimum_record), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [supremum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_supremum_record).

**gap lock: 间隙锁**

索引记录之间间隙（***gap***）上的锁，或第一个索引前或最后一个索引后的间隙上的锁。比如，SELECT c1 FOR UPDATE FROM t WHERE c1 BETWEEN 10 and 20；阻止其它事务向t.c1列中插入15，不管列中是否有这样的值，因为在范围中存在的值之间的间隙被锁住了。对比记录锁（***record lock***）与行间隙锁（***next-key lock***）。

间隙锁是性能与并发（***concurrency***）之间的取舍的一部分，在有些事务隔离级别（***isolation levels***）上是有用的，有些则不然。

参见 [gap](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap), [infimum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_infimum_record), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [next-key lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_next_key_lock), [record lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_record_lock), [supremum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_supremum_record).

**general log: 数据库日志**

参见 [general query log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_general_query_log).

**general query log: 数据库日志**

一种用来为MySQL服务处理过的SQL语句的诊断问题与排除故障的日志（***log***）。可以被存储在文件中或数据库表中。你必须通过general\_log配置选项使其生效来使用它。你可以用sql\_log\_off配置选项为一个指定的连接禁用它。

记录比慢查询（***slow log***）日志更广泛的查询。不像用来做镜像的二进制日志（***binary log***），数据库日志包括SELECT语句并且不会维护严格的顺序。如需更多信息，参见[第5.2.3节，数据库日志](https://github.com/xufubo/manual56/blob/master/docs/Chapter_05/05.02.03_The_General_Query_Log.md)。

参见 [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [general query log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_general_query_log), [log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log).

**global\_transaction: 全局事务**

分布式事务（***XA***）中的一种事务（***transactions***）。它由几个本身就是事务的动作组成，但这些事务又必须以一组的方式一起成功完成，或以一组的方式整体回滚。本质上来讲，这让ACID的属性拓展升级，为的是让多个ACID事务可以做为一个全局的同时拥有ACID属性的事务来执行。更多信息：

[ACID](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_acid), [transaction](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_transaction), [XA](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_xa)

**group commit: 组提交**

一个InnoDB的优化，以一次一组的提交方式执行一些低级I/O操作（***log write***），而不是单独为每次提交都做刷新与同步。

**hash index: 哈希索引**

一种用于等号运算符而不是范围运算符的查询的索引类型。它在MEMORY引擎表中有效。尽管哈希索引因为历史原因成为了MEMORY表的默认索引，但是这种存储引擎也支持B树（***B-tree***）索引，B树索引在通用的查询中常常是个不错的选择。MySQL中有一种这类索引的变体，自适应哈希索引（***apaptive has index***），它是在运行时按需为InnoDB表自动生成的。

**HDD: 机械磁盘或不译**

hard disk drive的首字母缩写。指的是使用旋转盘片的存储媒介，一般用来与固态硬盘（***SSD***）做比较与对照。

**heartbeat: 心跳**

为表示系统功能正常而周期性的发出的信息。在复制（***replication***）环境中，如果主库（***master***）停止发送此类信息，其中一个从库（***slave***）会替代它的位置。类似的技术可以被用到集群环境中的两台服务器之间，用来确保它们都正常运转。高可用的常用机制。

**high-water mark 高水位线**

一个表示上限的值，运行时不允许超越的硬性限制，要么是实际中到达的最大值。与之对应的是低水位线（low-water mark）。

**history list: 清除链表**

含有被标为删除的记录的事务（***transaction***）列表，这些记录会被InnoDB清除（***purge***）操作有计划地处理。记录在\*\*\*undo log\*\*\*。清除链表的长度可以在SHOW ENGINE INNODB STATUS命令的报表中看到。如果清除链表增长到超过[innodb\_max\_purge\_lag]配置选项的值，每个\*\*\*DML\*\*\*操作会被静默延迟，以让清除操作完成刷新（***flush***）删除的记录。

也叫 ***purge lag***.

参见 [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge), [purge lag](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge_lag), [rollback segment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback_segment), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**hot: 热**

一行. 一张表或内部数据结构被访问得非常频繁的一种情况，需要某些模式下的锁和互斥，它会带来性能与扩展性上的问题。

虽然“热”一般指的是不受欢迎的情况，但热备（***hot backup***）却是备份中的较好的选择。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup).

**hot backup: 热备**

种在数据库在运行且应用在读写它的情况下的备份。这种备份涉及的不是简单的拷贝文件：它必须包括任何在备份进行时插入或更新的数据；它必须排除掉任何在血仇进行时被删除的数据；它必须忽略任何没有提交的变更。

Orcale的执行热备的产品，尤其是对InnoDB表，还有对MyISAM或其它存储引擎表，叫MySQL企业备份（***MySQL Enterprise Backup***）。

热备过程由两个阶段组成。最初的拷贝数据生成一个原始备份（***raw backup***）。应用（***apply***）步骤合将任何在备份运行过程中发生的变更都合并到数据中。应用变更会生与一个一致备份（***prepared backup***）；这些文件已经为随时恢复做好准备。

参见 [apply](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_apply), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [prepared backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_prepared_backup), [raw backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_raw_backup).

**I**

**I/O-bound: I/O带宽**

参见[disk-bound](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_disk_bound)

**ib-file set: ib文件集**

MySQL数据库内部由InnoDB管理的一组文件：系统表空间（***system tablespace***）. 任何\*\*\*file-per-table\*\*\*表空间以及\*\*\*redo log\*\*\*文件（一般有两个）。为了避免不同DBMS产品之间对数据库（***database***）含义以及MySQL数据库中非InnoDB文件的部分产生歧义，有时用在InnoDB文件结构与格式的细节讨论上。

参见 [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace).

**ib\_logfile: redolog文件或不译**

构成\*\*\*redo log\*\*\*的一组文件，一般以ib\_logfile0和ib\_logfile1来命名。有时也被称为日志组（***log group***）。这些文件记录了尝试更改InnoDB表中数据的语句。在崩溃后的重启中，这些语句会自动重放到被未完成的事务写过的正确的数据中。

这些数据不能被用来做手动恢复；对于这种情况，使用二进制日志（***binary log***）。

参见 [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [log group](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log_group), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log).

**ibbackup\_logfile: 不译**

在热备（***hot bakcup***）操作中由MySQL企业备份（***MySQL Enterprise Backup***）产品创建的补充备份文件。它包含任何在备份运行时发生的变更的信息。这些包含ibbackup\_logfile的初始备份文件叫做原始备份（***raw backup***），因为在备份操作时间产生的变改尚未收入。当你向原始备份文件执行了应用（***apply***）步骤后，生成文件的确包含了那些最终的变量，这叫一执备份（***prepared backup***）。在这个阶段，ibbackup\_logfile就没有用了。

参见 [apply](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_apply), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [prepared backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_prepared_backup), [raw backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_raw_backup).

**.ibd file: .ibd文件**

每个使用\*\*\*file-per-table\*\*\*模式创建的InnoDB表（***table***）会进入到数据库目录下自己的表空间（***tablespace***）文中，以.ibd为后缀名。这个文件包含表的数据和表的任何索引。file-pre-table模式由\*\*\*innodb\_file\_per\_table\*\*\*选项来控制，影响InnoDB存储的使用量与性能，在MySQL 5.6.7及更高版本中默认开启。

这个后缀不适用于由ibdata文件（***ibdata files***）组成的系统表空间（***system tablespace***）。

当一个.ibd文件包含在MySQL企业备份（***MySQL Enterprise Backup***）产生的压缩备份中时，对应的压缩文是一个.ibz文件。

在MySQL 5.6或更高版本中，如果一个表使用DATA DIRECTORY ＝ 字句创建表时，.ibd文件会定位到正常数据库目录之外，并用一个.isl文件来指向它。

参见 [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [ibdata file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibdata_file), [.ibz file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibz_file), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [innodb\_file\_per\_table], [.isl file][isl file], [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**ibdata file: ibdata文件**

像ibdata1. ibdata2等等这样命名的一组文件，它们组成InnoDB的系统表空间（***system tablespace***）。这些文件包含InnoDB表的元数据（***data dictionary***，数据字典）和为\*\*\*undo log\*\*\*. 变更缓冲（***change buffer***）和双写缓冲（***doublewrite buffer***）等提供的存储区。它们也能包含一些或全部表的数据（取决于\*\*\*file-per-table\*\*\*模式是否开启）。当\*\*\*innodb\_file\_per\_table\*\*\*选项生效，新为新表创建的数据和索引存储在单独的\*\*\*.ibd\*\*\*文件中，而不在系统表空间中。

ibdata文件的增长受innodb\_autoextend\_increment配置选项的影响。

参见 [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [innodb\_file\_per\_table], [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**ibtmp file: ibtmp文件**

为非压缩InnoDB临时长或相关对象提供的InnoDB临时表空间数据文件。配置文件选项innodb\_temp\_data\_file\_path允许用户为临时数据文件定义对应的目录。如果innodb\_temp\_data\_file\_path没有定义，默认行为是在数据目录下的ibdata1文件边上创建一个名为ibtmp1的单独的数据文件，以12MB的大小自动递增。

参见 [temporary tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_temporary_tablespace).

**.ibz file: .ibz文件**

当MySQL企业备份产品（***MySQL Enterprise Backup***）执行一下压缩备份（***compressed backup***）时，它将每个由\*\*\*file-per-table\*\*\*选项产生的后缀为.ibd的表空间转换为后缀为.ibz的文件。

在备份过程中应用的压缩与在正常操作中保持数据压缩的压缩行格式（***compressed row format***）是不一样的。一个压缩备份操作会忽略掉那些已经是压缩行格式表空间的压缩步骤，二次压缩会延缓备份却很少带来空间上的节省。

参见 [compressed backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_backup), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), tablespace.

**ilist: 索引词链表**

InnoDB全文索引（***FULLTEXT index***）中，由文件编号和位置信息的标号（***token***）（也就是一个特定的词）组成的数据结构。

参见 [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**implicit row lock: 隐式行锁**

一个为了确保数据一致性的InnoDB的行级锁，你不需要特别地请求它。

参见 [row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_lock).

**in-memory database: 内存数据库**

一种为了避免磁盘I/O负载和磁盘块与内存区之间传输负载而在内存中维护数据的一种数据库系统。一些内存数据库牺牲了持久性（ACID设计理念中的“D”），易受硬件. 电源和其它类型的故障的影响，使得它们更适合于只读操作。

MySQL features that are address the same kinds of memory-intensive processing include the InnoDB buffer pool, adaptive hash index, and read-only transaction optimization, the MEMORY storage engine, the MyISAM key cache, and the MySQL query cache.

MySQL处理内存密集型进程的特性包括InnoDB ***buffer pool***. 自适应哈希索引（***adaptive hash index***）. 只读事务（***read-only transaction***）优化. MEMORY存储引擎. MyISAM key cache以及MySQL查询缓存（***query cache***）。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [adaptive hash index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_adaptive_hash_index), [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [disk-based](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sk_based), [read-only transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_only_transaction).

**incremental backup: 增量备份**

一种热备（***hot backup***），由MySQL企业备份（***MySQL Enterprise Backup***）执行，它只保存某个时间点后改变了的数据。一个全量备份和一连串的增量备份可以让你在一个很长的时间段里重构你的备份数据，而没少手头上要保留多个全备份的存储压力。你可以先恢复全备份，然后连续应用每个增量备份，也可以应用每个增量备份让全备份保持更新状态，然后再执行一个单独的恢复操作。

The granularity of changed data is at the page level. A page might actually cover more than one row. Each changed page is included in the backup.

变更了的数据的粒度在页（***page***）层级。一个页可能实际上覆盖不至一行。每个变更了的行都包含在备份中。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**index: 索引**

一种数据结构，提供快速查找表（***table***）中行（***row***）的功能，通常是生成一棵树结构（***B-Tree***），代表了一个特定列或一组列中的值。

InnoDB表一直拥有一个相当于主键（***primary key***）的簇索引（***clustered index***）。它们可以拥有定义在一列或多列上的一个或多个二级索引（***secondary key***）。依据二级索引的结构，它们可以划分为部分索引（***partial index***）. 单列索引（***column index***）和组合索引（***composite index***）。

The ideal database design uses a covering index where practical;.

索引是查询（***query***）性能的一个关键因素。数据库架构师设计表. 查询和索引以允许快速查找应用程序所需要的数据。理想的数据库设计使用实用的覆盖索引（***covering index***）；查询结果可以完全从索引中计算出来，而不用读实际的表数据。为了高效地检查值是否在父表（***parent table***）和子表（***child table***）中存在，每个外键约束（***foreign key***）也需要索引。

尽管B树索引最常用，但另一种数据结构被用来做哈希索引（***hash index***），在MEMORY存储引擎和InnoDB自适应索引（***adaptive hash index***）中。

参见 [adaptive hash index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_adaptive_hash_index), [B-tree](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_b_tree), [child table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_child_table), [clustered index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clustered_index), [column index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_column_index), [composite index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_composite_index), [covering index], [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key), [hash index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hash_index), [parent table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_parent_table), [partial index], [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [row](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**index cache: 索引缓存**

为InnoDB全文搜索（***full-text search***）保存令牌数据的内存区。它在当是全文索引（***FULLTEXT index***）列中的数据被删除或更新时将该数据缓冲起来以减小磁盘I/O。在索引缓存变满时令牌数据会被写到磁盘上。每个InnoDB的全文索引都有各自独立的索引缓存，它的大小由配置选项innodb\_ft\_cache\_size控制。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**index hint: 索引提示**

一个为了覆盖优化器推荐索引（***index***）使用的扩展SQL语法。例如，FORCE INDEX. USE INDEX和IGNORE INDEX字句。通常是在索引列上的值分布不均匀，导致基数（***cardinality***）评估不准确的情况下使用。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index).

**index prefix: 索引前缀**

在一个多列索引（***composite index***，组合索引）中，索引（***index***）的初始列或前导列。一个引用组合索引中最前面的1. 2. 3等等列的查询可以使用到索引，即使这个查询没有引用到索引中的所有列。

参见 [composite index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_composite_index), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index).

**index statistics: 索引统计**

参见 [statistics].

**infimum record: 下确界记录**

索引（***index***）中假想的记录（***psesudo-index***），代表这个索引中低于最小值的间隙（***gap***）。如果一个事务有一个诸如SELECT ... FOR UPDATE ... WHERE col < 10;的语句；并且这个列中最小值为5，它将锁住这个下确界记录以阻止其它事务插入更小的值，比如0啊. －10啊等等。

参见 [gap](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [pseudo-record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_pseudo_record), [supremum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_supremum_record).

**INFORMATION\_SCHEMA: 系统信息库**

为MySQL数据字典（***data dictionary***）提供查询接口的数据库的名字。（此名由ANSI SQL标准定义。）为了检查数据库的信息（元数据），你可以查询诸如INFOMATION\_SCHEMA.TABLES和INFOMATION\_SCHEMA.COLUMNS的表，而不使用产生没有结构化输出的SHOW命令。

信息系统库包含一些\*\*\*InnoDB\*\*\*特定的表，如INNODB\_LOCKS和INNODB\_TRX。你使用这些表不是用来看这个数据库是如何组织的，而是获取关系InnoDB表的实时信息来帮助做性能监控. 调优和故障排除的。实际上，这些表提供了关于MySQL压缩（***compression***）. 事务（***transaction***）以及它们对应锁（***lock***）的特性的数据。

参见 [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**InnoDB**

一个MySQL组件，它高性能与事务（***transaction***）功能结合起来，提供可靠性. 鲁棒性与并发访问。它体现了\*\*\*ACID\*\*\*的设计理念。表示一个存储引擎（***storage engine***）；它可以用ENGINE=INNODB子句来处理表的创建与更改。架构细节和管理程序参见[第14.2节，InnoDB存储引擎][14.02.00]，性能建议参见[第8.5节，优化InnoDB表][08.05.00]。

在MySQL 5.5或更高版本中，InnoDB是新创建表的默认存储引擎，ENGINE=INNODB子句不是必需的了。仅在MySQL 5.1中，InnoDB的很多优势特性需要启用InnoDB Plugin组件。要考虑过渡到InnoDB表是默认的最近的版本，请参见[第14.2.1.1节，InnoDB作为默认的存储引擎][14.02.01.01]。

InnoDB表非常适合热备（***hot backup***）。要了解MySQL企业备份产品（***MySQL Enterprise Backup***）在不打断正常进程的情况下备份MySQL服务的信息，请参见[第24.2节，MySQL企业备份][24.02.00]。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [storage engine](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_storage_engine), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**innodb\_autoinc\_lock\_mode: innodb参数，不用译**

innodb\_autoinc\_lock\_mode控制自增锁（***auto-increment locking***）所使用的算法。当你有一个自增主键（***primary key***）时，在设置innodb\_autoinc\_lock\_mode=1的情况下，你只能使用基于语句的复制。这个设置也叫连续锁模式，因为一个事务中的多行插入会接收到连续的自增值。如果你使用了允许更高并发的插入操作innodb\_autoinc\_lock\_mode=2，就要使用基于行的复制，而不是基于语句的复制。这个设计叫交错（***interleaved***）锁模式，因为同一时间运行的多个多行插入可以接到交错的自增值。设置innodb\_autoinc\_lock\_mode=0是之前的（传统）的默认设置，在处理兼容时可能会被用到。

参见 [auto-increment locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_auto_increment_locking), [mixed-mode insert](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mixed_mode_insert), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key).

**innodb\_file\_format: innodb参数，不用译**

在你指定了innodb\_file\_format选项的一个值以后，所有创建InnoDB表空间的文件格式（***file format***）都取决于这个选项。要创建不同于系统表空间（***system tablespace***）的表空间（***tablespace***），你也必须使用\*\*\*file-per-table\*\*\*选项。目前，你可以指定\*\*\*Antelope\*\*\*和\*\*\*Barracuda\*\*\*文件格式。

参见 [Antelope](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_antelope), [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [file format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_format), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [innodb\_file\_per\_table], [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**innodb\_file\_per\_table: innodb参数，不用译**

这是一个影响InnoDB文件存储. 功能可用性及I/O好多方面的非常重要的配置。在MySQL 5.6.7之前，它是默认关闭的。innodb\_file\_per\_table选项打开了file-per-table模式，它将每个新创建的InnoDB表和关联的索引存储在系统表空间（***system tablespace***）之外自己的.ibd文件（\*\*\*.ibd file\*\*\*）中。

这个选项会影响到一些SQL语句的性能与存储方面的考虑，比如DROP TABLE和TRUNCATE TABLE。

很多InnoDB的其它特性也需要它来充分发挥优势，比如表压缩（***compression***）或用MySQL企业备份（***MySQL Enterprise Backup***）来备份指定的表。

这个选项曾经是静态的，不过现在可以用SET GLOBAL命令来设置了。

如需参考信息，参考[innodb\_file\_per\_table]。如需使用信息，参见[第14.2.6.2节，InnoDB File-Per-Table模式][14.02.06.02]。

参见 [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace).

**innodb\_lock\_wait\_timeout: innodb参数**

innodb\_lock\_wait\_timeout选项在等待被共享的资源变得可用与放弃并在你的应用中处理错误. 重试或做替代处理二者之间设置一个平衡。任何等待获取锁超过指定时间的InnoDB事务都会被回滚。这在处理由更新多个不同存储引擎表时引起的死锁时非常有用；这类死锁是没法自动检测的。

参见 [deadlock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_deadlock), [deadlock detection](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_deadlock_detection), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [wait](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_wait).

**innodb\_strict\_mode**

The innodb\_strict\_mode option controls whether InnoDB operates in strict mode, where conditions that are normally treated as warnings, cause errors instead （and the underlying statements fail）. innodb\_strict\_mode选项控制InnoDB的操作是否在严格模式下，那些在普通模式下被视为警告的问，在严格模式下都会用错误来代替（及相关的语句错误）。该模式在MySQL 5.5.5及更高版本中是默认设置。

**insert: 插入**

SQL中一个主要的DML操作。如果插入性能对你很重要，你应该了解InnoDB的功能，如：***change buffering***中使用的**insert buffering**和 **auto-increment等**。更多信息：

 [auto-increment](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_auto_increment), [change buffering](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_change_buffering), [data warehouse](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_data_warehouse), [DML](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_dml), [InnoDB](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_innodb), [insert buffer](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_insert_buffer), [OLTP](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_oltp), [SQL](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_sql)

**insert buffer**

***change buffer***以前的叫法。现在的***change buffering***包括删除缓冲. 更新缓冲以及插入缓冲。

**insert buffering**

一种将由INSERT操作引起的二级索引（secondary index也叫辅助索引）的变更存储在***insert buffer***中的技术，而不是直接将它们写到磁盘，目的是减少随机I/O。它是***change buffering***的一种；其它的为删除缓冲技术（***delete buffering***）与清除缓冲技术（***purge buffering***）。如果二级索引（secondary index也叫辅助索引）为唯一（***unique***）索引，**insert buffering**是无用的，因为新值的唯一性在新的实体写入磁盘之前是无法验证的，因此对于唯一索引这种方法就不适用了。其它的***change buffering***对唯一索引有效。更多信息：

 [change buffer](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_change_buffer), [change buffering](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_change_buffering), [delete buffering](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_delete_buffering), [insert buffer](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_insert_buffer), [purge buffering](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_purge_buffering), [unique index](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_unique_index)

**instance: 实例**

一个单独的mysqld后台进程（***daemon***），管理包含一个或多个数据库的一个数据目录（***data directory***），每个数据库都包括一组表（***table***）。在开发. 测试和一些复制（***replication***）场景下，一台机器多个实例是很正常的事，每个实例管理自己的数据目录，监听各自的端口或套接字。

**instrumentation: 监测**

为调优与调试收集性能数据而在源码级做的修改，通过监测收集到的数据使用INFORMATION\_SCHEMA和PERFORMANCE\_SCHEMA数据库通过SQL接口暴露出来。

参见 [INFORMATION\_SCHEMA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_information_schema), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema).

**intention exclusive lock: 意向排它锁**

参见 [intention lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_intention_lock).

**intention lock 意向锁**

一种表级锁，常常指的是那种事务试图在表行上获取的锁。不同在事务可以在同一张表上获取到不同的意向锁，但第一个事务在表上获取一个意向排它锁（***intention exclusive***）以阻止其它事务再在该表上获取任何共享锁或排它锁。相反地，第一个事务在表上获取一个意向共享锁（***intention shared***）以阻止其它事务再在该表上获取任何排它锁。两阶段进程允许锁请求按序解决，而不用阻塞锁和兼容的相应的操作。如需各多此类锁机制的详情，请参考[第14.2.2.3节，InnoDB锁模式][14.02.02.03]。

参见 [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [lock mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock_mode), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking).

**intention shared lock: 意向共享锁**

参见 [intention lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_intention_lock).

**inverted index: 倒排索引**

为文档索引系统优化的数据结构，用在InnoDB全文检索（***full-text search***）实现中。InnoDB全文索引（***FULLTEXT index***），以倒排索引的方式实现，记录每个词在文档中的位置，而不是表中行的位置。单列的值（做为字符串存储的一个文档）可以由多个反向索引中的多个实体所表示。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index), [ilist](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ilist).

**IOPS: 每秒读写次数**

每秒读写次数（***I/O operations per second***）的首字母缩写。繁忙系统，特别是OLTP应用系统的一个通用测量标准。如果这个值接近存储设备可以处理的最大值，该应用会变成磁盘受限（***disk-bound***），扩展性（***scalability***）也受到限制。

**.isl file: .isl文件**

在MySQL 5.6及更高版本中，一个用来指定由用DATA DICRECTORY =字句创建的InnoDB表的.ibd文件（\*\*\*.ibd file\*\*\*）位置的文件。它的作用就像符号链接，只是没有实际符号链接机制中平台限制。你可以在数据库（***database***）目录之外存储InnoDB表空间（***tablespace***），例如，根据表的用途，可以放在特别大或特别快的存储设备上。如需更多详情，参参[第14.2.6.4节，指定表空间的位置][14.02.06.04]。

参见 [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**isolation level: 事务隔离级别**

数据库进程基础之一。隔离性是缩写\*\*\*ACID\*\*\*中的I；隔离级别是在多个事务（***transaction***）在同一时间产生变更与执行查询的情竞下，对性能与可靠性. 一致性和结果再生性之间进行微调的设置。

一致性与保护制度从高到低，InnoDB所支持的隔离级别为：***SERIALIZABLE***. ***REPEATABLE READ***. \*\*\*READ COMMITTED\*\*\*和\*\*\*READ UNCOMMITTED\*\*\*。

对于InnoDB表，很多用户可以为所有的操作保留默认的隔离级别（***REPEATABLE READ***）。专家用户可以选择\*\*\*read committed\*\*\*级别，因为他们追求OLTP进程的扩展性极限，或在数据仓库操作中较小的不一致不会影响到大量数据的聚合结果。两头的级别（\*\*\*SERIALIZABLE\*\*\*和\*\*\*READ UNCOMMITTED\*\*\*）改变处理的行为的程度很大，所以很少用它们。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [READ COMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_committed), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**join: 关联**

通过引用表中具有相同值的列来从多个表检索数据的查询（***query***）。理想情况下，这些列是InnoDB外键（***foreign key***）关系中的一部分，它们确保了参照完整性（***referential intgrity***），并且关联的列上是有索引（***index***）的。通过范式设计用来提高空间和查询性能。更多信息：

[foreign key](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_foreign_key), [index](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_index), [normalized](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_normalized), [query](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_query), [referential integrity](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_referential_integrity)

**KEY\_BLOCK\_SIZE: InnoDB表选项**

用来指定使用了压缩行格式（***compressed row format***）的InnoDB表中数据页大小的选项。默值为8KB。越小的值达到内部限制的风险越高，该限制依赖于行的大小和压缩比。

**latch: 闩锁**

InnoDB针对自己内部内存结构体实现锁（***lock***）的一个轻量级的结构体，通常保持毫秒或微秒级的短暂的时间。包含互斥（***mutex***，对于排它访问）和读写锁（***rw-lock***，对于共享访问）一般术语。某些闩锁的重点在性能调优上，比如数据字典（***data dictionary***）互斥。对于锁使用和竞争的统计可以通过Performance数据库（***Performance Schema***）接口获得。

参见 [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema), [rw-lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rw_lock).

**list: buffer页面lru链表**

InnoDB buffer pool相当于一个内部页面的LRU链表。

**lock: 锁**

控制访问诸如表. 行或内部数据结构等资源的对象的高级的概念，是锁（***locking***）机制中的一部分。针对进一步的性能调优，你可以探索实现了锁的真正的结构体，如互斥锁和闩锁（***latch***）。

参见 [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [lock mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock_mode), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex).

**lock escalation 锁晋级**

某些数据库系统使用的一种操作，将多行的锁合并为一个单独的表锁，节省内存空间，但降低对表的并发访问。InnoDB针对行锁使用一个空间高效的方式，所以不需要锁晋级。

参见 [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_lock), [table lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table_lock).

**lock mode: 锁模式**

一个共享锁（S）允许事务读一行。多个事务可以在同一时间在同一行上获取一个S锁。

一个排它锁允许事务更新或删除一行。其它的事务在同一时刻不再能获得任何类型的锁。

意向锁适用于表级，并且常常指的是事务试图在表行上获取的那种锁。不同在事务可以在同一张表上获取到不同的意向锁，但第一个事务在表上获取一个意向排它锁（***intention exclusive***）以阻止其它事务再在该表上获取任何共享锁或排它锁。相反地，第一个事务在表上获取一个意向共享锁（***intention shared***）以阻止其它事务再在该表上获取任何排它锁。两阶段进程允许锁请求按序解决，而不用阻塞锁和兼容的相应的操作。

参见 [intention lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_intention_lock), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking).

**locking: 锁机制**

防止一个事务（***transaction***）查询或变更正在被其它事务查询或变更的数据的系统。锁策略一定要衡量数据库操作的可靠性及一致性（\*\*\*ACID\*\*\*理念的原则）与良好的并发（***concurrency***）所需的性能之间关系。微调锁策略往往会涉及选择一个隔离级别（***isolation level***）并保证你所有的数据库操作对于该隔离级别都是安全的和可靠的。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**locking read: 加锁读**

在一个InnoDB表上同时执行了加锁（***locking***）操作的SELECT语句。无论是SELECT ... FOR UPDATE或是SELECT LOCK IN SHARE MODE。取决于事务的隔离级别（***isolation level***），它有产生死锁（***deadlock***）的潜在可能性。与它对应的是无锁读（***non-locking read***）。在只读事务中不允许对全局表做此类操作。

参见 [deadlock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_deadlock), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [non-locking read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_non_locking_read), [read-only transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_only_transaction).

**log: 日志**

在InnoDB语境中，“log”和“log files”通常指的是表示为ib\_logfile\*文件的redo日志（***redo log***）。另一个log区是undo日志（***undo log***），它是物理系统表空间（***system tablespace***）的一部分。

其它MySQL很重要的日志就是错误日志（***error log***，用来诊断启动与运行时错误）. 二进制日志（***binary log***，用来做复制和执行定点恢复）. 数据库常规日志（***general query log***，用来诊断应用错误）和慢查询日志（***slow query log***，用来诊断性能问题）。

参见 [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [error log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_error_log), [general query log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_general_query_log), [ib\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ib_logfile), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [slow query log], [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**log buffer: 日志缓冲**

一个内存区，保存要写到由redo日志（***redo log***）组成的日志文件（***log file***）中的数据。由配置选项[innodb\_log\_buffer\_size]控制。

参见 [log file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log_file), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log).

**log file: 日志文件**

组成redo日志的ib\_logfileN中的一个文件。数据从日志缓冲内存区写入到这些文件中。

参见 [ib\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ib_logfile), [log buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log_buffer), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log).

**log group: 日志组**

组成redo日志（***redo log***）的文件集合，通常以ib\_logfile1和ib\_logfile2。（由于这个原因，有是统称为\*\*\*ib\_logfile\*\*\*。）

参见 [ib\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ib_logfile), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log).

**logical: 逻辑**

涉及高层的. 诸如表. 查询. 索引以及其它SQL概念的抽象角度的操作。通常地，逻辑角度对于数据库管理和应用部署的易用性与实用性都是非常重要的。与它对应的是物理（***physical***）。

参见 [logical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_logical_backup), [physical](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical).

**logic backup: 逻辑备份**

一个不用拷贝实际的数据文件，重新生成表结构与数据的备份（***backup***）。例如，mysqldump命令会产生一个逻辑备份，因为它的输出包含诸如可以重建表的CREATE TABLE和INSERT语句。与它相应的是物理备份（***physical backup***）。逻辑备份提供了灵活性（比如，你可以在恢复前修改表的定义或插入语句），但要比物理备份花销更长的时间。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [mysqldump](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqldump), [physical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical_backup), [restore](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_restore).

**loose\_: 一个前缀，不译**

在MySQL 5.1中，在服务启动后安装InnoDB Plugin时，加在InnoDB配置选项前的一个前缀，这样避免了任何当前版本中没有注册过的新配置选项所导致的启动失败。如果该前缀后面的部分不是一个注册过的选项，MySQL在处理此前缀开始的配置选项时，会给出一个警告，而不是启动失败。

参见 [plugin](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_plugin).

**low-water mark: 下限**

一个代表着低值的限制的数值.一般来说,是一个阈值,该阈值的作用是一旦达到这个阈值,一些纠错动作会开始或者变得更活跃。与它对应的是上限（***high-water mark***）。

参见 [high-water mark](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_high_water_mark).

**LRU: 最近最少使用**

最近最少使用（***least recently used***）的首字母缩写，一个管理存储空间的常用方法。当空间中需要缓存新的项目时，那些最近没有被使用到的项目会被驱逐出去。InnoDB默认使用LRU机制管理\*\*\*buffer pool\*\*\*中的页（***page***），但在一个页可能只读一次的情况下会有个例外，比如在一次全表扫描（***full table scan***）过程中。LRU算法的变体叫中值插入策略。buffer pool的管理方法与传统的LRU算法不同之处在于它是通过[innodb\_old\_blocks\_pct]. [innodb\_old\_blocks\_time]和MySQL 5.6中的选项[innodb\_lru\_scan\_depth]及[innodb\_flush\_neighbors]来微调的。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [eviction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_eviction), [full table scan](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_table_scan), [midpoint insertion strategy](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_midpoint_insertion_strategy), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**LSN: 日志序号**

日志序列号的首字母缩写。这个单调递增的值代表了redo日志中所记录的操作所对应的时间点。（这个时间点不会理会事务边界；它可以落在一个或多个事务的中间。）它用在InnoDB内部的崩溃恢复和管理buffer pool中。

在MySQL 5.6.3之前，LSN是一个4字节无符号整型值，在MySQL 5.6.3中，当redo日志的大小限制从4GB变到512GB时，LSN变成8字节无符号整型值，附加的字节被用来存储额外的大小信息。在MySQL 5.6.3或更晚的版本上编译的应用可能会用到64位，而不是32位变更来存储和比较LSN的值。

在MySQL企业备份产品中，你可以指定一个相当于时间点的LSN来生成一个增量备份。相关的LSN会在mysqlbackup命令的输出中显示。一旦你有了一个全备的相当于时间点的LSN，你就可以指定一个值来生成一个后续的增量备份，它的输入中包含另一个下次增量备份的所需的LSN。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [incremental backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_incremental_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**M**

**master server: 主服务器**

经常简称为“master”。复制（***replication***）环境中的一个数据库服务机器，用来处理对数据的最初的插入. 更新和删除请求。这些数据被传送到并在其上重复执行的其它服务器叫做从服务器（***slave servers***）。

参见 [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication), [slave server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slave_server).

**master thread: 主线程**

一个在后台执行各种任务的InnoDB线程（***thread***）。大多数情况是I/O相关的，比如从插入缓冲（***insert buffer***）中往相关的二级索引中写入变更。

为了提高并发（***concurrency***），有时候行为会从主线程转移到单独的后台线程中。比如，在InnoDB 5.6及更高版本中，脏页（***dirty pages***）会由页清理器（***page cleaner***）从\*\*\*buffer pool\*\*\*中刷新，而不是由主线程执行。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [insert buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffer), [page cleaner](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page_cleaner), [thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_thread).

**MDL: 元数据锁**

metadata lock的简称。

参见 [metadata lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_metadata_lock).

**memcached: 不译**

一个众多MySQL和\*\**NoSQL****软件栈中流行的组件，允许快速读写单值并将结果完整地缓存在内存中。传统上，应用需要额外的逻辑往MySQL数据库中写入相同的数据来达到持久化，或者在数据还没有在内存中缓存时需要从MySQL数据库中读取数据。现在，应用可以使用简单的****memcached*协议来直接使用\*\**InnoDB****或MySQL集群的表来与MySQL服务通信了，大多数据语言都有支付该协议的客户端。这些MySQL表的NoSQL接口允许应用达到比直接使用SQL命令更高的读写性能，并且可以简化那些已经为内部缓存收纳了****memcached*的系统的应用逻辑与部署配置。

InnoDB表的memcached接口在MySQL 5.6及更高版本中可能；参考[第14.2.17节，InnoDB集成memcached][14.02.17]；细节参考[[http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.html]。](http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.html%5D%E3%80%82)

参见 [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [NoSQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_nosql).

**merge: 合并**

将变更应用到内存中缓存的数据上，比如当一个页被加载到\*\*\*buffer pool\*\*\*中时，以及任何在变更缓冲（***change buffer***）中记录的可应用的变更纳入进buffer pool中。更变过的数据最终由刷新（***flush***）机制写入到表空间（***tablespace***）中。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**metadata lock: 元数据锁**

一种锁（***lock***）类型，为了防止同一时间另一事务（***transaction***）正在使用表时的\*\*\*DDL\*\*\*操作。如需更多细节，参考[第8.10.4节，元数据锁][08.10.04]。

特别是在MySQL 5.6及更高版本中，对在线（***online***）操作的改进都集中在减少元数据锁的量上。目的是当某表被其它事务查询以及更新等时，对于不改变表结构的DDL操作（诸如针对InnoDB表的CREATE INDEX和DROP INDEX操作）可以继续。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [online](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**metrics counter: 计数器**

在MySQL 5.6及更高版本中，一个由information\_schema库中innodb\_metrics实现的特性。你可以查询低级别的InnoDB操作的次数（***counts***）和总数，以及使用这些结果结合\*\*\*performance\_schema\*\*\*库中的数据来进行性能调优。

参见 [counter](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_counter), [INFORMATION\_SCHEMA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_information_schema), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema).

**midpoint insertion strategy: 中间插入策略**

是将最初的页不加载到InnoDB \*\*\*buffer pool\*\*\*中“最新”列表的最后，而是放在中间某个位置的技术。具体的位置点可以不同，基于[innodb\_old\_blocks\_pct]选项的设定。意图是只读 一次的块，诸如在全表扫描（***full table scan***）过程中，可以比根据严格的\*\*\*LRU\*\*\*算法更快地老化出buffer pool。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [full table scan](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_table_scan), [LRU](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lru), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**mini-transaction: 迷你事务**

在\*\*\*DML\*\*\*操作中，在物理（***physical***）层对内部数据结构进行变更时InnoDB所处理的一个内部阶段。迷你事务没有回滚（***roll back***）的概念；多个迷你事务可以发生在一个事务（***transaction***）中。迷你事务的信息写入到崩溃恢复（***crash recovery***）期间使用的redo日志（***redo log***）中。迷你事务也可以发生在常事务环境之外，例如在后台线程所处理的清除（***purge***）操作期间。

参见 [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [physical](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**mixed-mode insert: 混合模式插入**

一个对新行指定了部分但没有指定所有自增（***auto-increment***）值的INSERT语句。例如，一个多值INSERT可能在一些情况下为自增列指定值，在别的情况下为NULL值。InnoDB在自增列被指定为NULL时会生成自增值。另一个例子是INSERT ... ON DUPLICATE KEY UPDATE语句，其中自增值可以生成但用不到，任何重复的行都会以UPDATE语句来处理，而不是以INSERT语句。

可以导致复制配置中的主库和从部数据不一致。可以要求调整innodb\_autoinc\_lock\_mode配置选项的值。

参见 [auto-increment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_auto_increment), [innodb\_autoinc\_lock\_mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb_autoinc_lock_mode), [master server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_master_server), [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication), [slave server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slave_server).

**.MRG file: .MRG文件**

MERGE存储引擎使用的一个文件，包含对其它表的引用情况。此后缀的文件总是包含在由MySQL企业备份产品（***MySQL Enterprise backup***）中mysqlbackup命令（***mysqlbackup command***）所生成的备份中。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command).

**multi-core: 多核**

可以利用多线程程序（如MySQL服务）优势的一种处理器。

**multiversion concurrency control: 多版本并发控制**

见 [MVCC](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mvcc).

**mutex: 互斥**

“互斥量（***mutex variable***）”的非式缩写。（Mutex自身是mutual exclusion的缩写。）是InnoDB对内部内存中的数据结构体用来表示或强制排它访问锁（***lock***）的低级对象。当这个锁被获取，任何其它的进程. 线程以及其它都被阻止获取相同的锁。与之相对的读写锁（***rw-lock***），是允许共享访问的。互斥与读写锁一并称为闩锁（***latch***）。

参见 [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [lock,] [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema), [Pthreads](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_pthreads), [rw-lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rw_lock).

**MVCC: 多版本并发控制**

“multiversion concurrency control”的首字母缩写。这种技术让InnoDB事务可以在一定的隔离级别（***isolation levels***）写执行一致性读（***consistent read***）操作；也就是说，查询那些正在被其它事务更新的行，并且能看到那些更新发生前的值。这对于提高并发来说是一个强大的功能，它在不用等待其它事务持有的锁的情况下允许查询继续执行。

这种技术在数据库界并不常见。一些数据库产品，以及一些其它的MySQL存储引擎都不支持它。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**my.cnf: 配置文件（Unix/Linux）**

Unix或Linuxt系统下MySQL配置文件名。

参见 [my.ini], [option file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_option_file).

**my.ini: 配置文件（Windows）**

Windows系统下MySQL配置文件名。

参见 [my.cnf], [option file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_option_file).

**.MYD file: .MYD文件**

MySQL用来存储MyISAM表数据的文件。

参见 [.MYI file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_myi_file), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command).

**.MYI file: .MYI文件**

MySQL用来存储MyISAM表索引的文件。

参见 [.MYD file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_myd_file), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command).

**mysql: mysql（客户端）**

mysql程序是MySQL数据库的命令行解释器。它通过请求\*\*\*mysqld\*\*\*后台驻留程序来处理\*\*\*SQL\*\*\*语句和一些MySQL特定的命令，诸如SHOW TABLES。

参见 [mysqld](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqld), [SQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sql).

**MySQL Enterprise Backup: MySQL企业备份**

一个执行MySQL数据库热备（***hot backup***）的授权产品。它提供最高效可靠的\*\*\*InnoDB\*\*\*表备份，但也能备份MyISAM和其它类型的表。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb).

**mysqlbackup command: mysqlbackup命令**

一个MySQL企业备份（***MySQL Enterprise Backup***）的命令行工具。它对InnoDB表执行热备（***hot backup***）操作，并对MyISAM和其它类型表执行温备（***warm backup***）操作。如需更多关于该命令行的信息，请参考[第24.2节，MySQL企业备份][24.02.00]。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [warm backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_warm_backup).

**mysqld: MySQL daemon（Unix）或MySQL service（Windows）**

mysqld程序是MySQL数据库的数据库引擎。它以Unix后台驻留程序或Windows服务的型式运行，在后台持续等待请求并执行维护工作。

参见 [mysql](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql).

**mysqldump: mysqldump命令**

一个执行数据库. 表和表数据混合体逻辑备份（***logical backup***）的命令。结果是可以重现原始结构对象. 数据或两都的SQL语句。对于大量的数据，像MySQL企业备份（***MySQL Enterprise Backup***）这样的物理备份（***physical backup***）解决方案是非常快速的，特别是对于恢复（***restore***）操作来说。

参见 [logical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_logical_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [physical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical_backup), [restore](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_restore).

**N**

**natural key: 自然主键**

一个索引列，通常是主键（***primary key***），其中的值是有现实意义的。通常不建议这么做，因为：

* 如果值万一变了，会有许多潜在的索引维护工作来重排簇索引（***clustered index***），并且更新在每一个二级索引（***secondary index***）里重复出现的主键拷贝。
* 即使看似稳定的值也会以一种不可预测的方式改变，很难在数据库中正确地表示改变。举个例子，一个国家可以变成两个或更多，这会让之前的国家码过时。或者，唯一值的规则可能发生异常。举个例子，即使纳税人的身份证号的初衷是一人一个，但数据库可能需要处理违犯规则的记录，如身份证被盗这种事儿。纳税人的身份证和其它敏感ID也很少做为主键，因为它们需要被保护. 加密以及其它不同于它列的对待。

因此，使用任意数字来组成一个人造（***synthetic key***）的键通常来说是更好的选择，比如使用自增列（***auto-increment***）。

参见 [auto-increment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_auto_increment), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index), [synthetic key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_synthetic_key).

**neighbor page: 相邻页**

相同簇（***extend***）中的任何一个页。当一个页（***page***）被选中用来刷新（***flush***）时，作为传统硬盘的I/O优化器会将任何相邻的脏页（***dirty page***）通常也被刷新。在MySQL 5.6及更高版本中，该行为可以被配置变量[innodb\_flush\_neighbors]所控制；你可以为SSD磁盘关掉该选项，它在写更小批量数据的随机查找上不会产生相同的负载。

参见 [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [extent](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_extent), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**next-key lock: 行间隙锁**

一个索引记录（***index record***）上记录锁和索引记录前间隙上间隙锁（***gap lock***）的组合。

参见 [gap lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [record lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_record_lock).

**non-blocking I/O: 同AIO**

专业术语，异步I/O（***asynchronous I/O***）的同义词。

参见 [asynchronous I/O](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_asynchronous_io).

**non-locking read: 不加锁读**

没有使用SELECT ... FROM UPDATE或SELECT ... LOCK IN SHARE MODE子句的查询（***query***）。在只读事务（***read-only transaction***）中全局表所允许的唯一的查询类型。与之相对的是加锁读（***locking read***）。

参见 [locking read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking_read), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [read-only transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_only_transaction).

**non-repeatable read: 非重复读**

这种情况下，一个查询检索了数据，随后一个查询在同一个事务（***transaction***）中检打算检索相同的数据，但是却返回了不同的结果（期间被另一个事务的提交所变更）。

这种操作类型与数据库设计的\*\*\*ACID\*\*\*理念相违背。在一个事务中，数据应该是一致的，有可预知和稳定的关系。

Among different isolation levels, non-repeatable reads are prevented by the serializable read and repeatable read levels, and allowed by the consistent read, and read uncommitted levels.

在不同的隔离级别（***isolation levels***）中，非重复读可以通过可序列化读（***serializable read***）和可重复读（***repeatable read***）来避免，在一致性读（***consistent read***），也就是未提交读（***read uncommitted***）级别中是允许的。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**normalized: （符合）范式的**

一个数据库设计策略，其中数据分拆到多个表中，并且重复的值简化为由一个ID代表的行，来避免存储. 查询和更新冗余或冗长的值。它通常用联机事务处理（***OLTP***）应用中。

举个例子，一个地址可能会给定一个唯一ID，这样一个普查的数据库可以通过将一个家庭中的每个成员与地址ID关联在一起来表现住在该地址下（***lives at this address***）的关系，而不用存储一个复杂值的多个拷贝，如中国北京前门外大街皮条胡同（***123 Main Street, Anytown, USA***）。

再举个例子，虽然一个电话本应用会把每个人的名字和地址连同电话号码存到同一张表中，但一个电话公司的数据库可能会给每个电话号码一个指定的ID，并将号码和ID存到一个独立的表中。这种范式的表现方式可以简化在区号分拆时的大范围的更新。

范式化也不总是推荐使用。主要用来查询且只在完全删除和加载时被更新的数据常常保存在更少. 更大的表中，重复的值存在冗余拷贝。这种数据表现方式叫反范式（***denormalized***），在数据仓库应用中十分常见。

参见 [denormalized](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_denormalized), [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key), [OLTP](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_oltp), [relational](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_relational).

**NoSQL: NoSQL**

表示一组数据访问技术的广泛的术语，它不使用\*\*\*SQL\*\*\*语句作为它们读写数据的主要途径。一些NoSQL技术是键-值存储，只接受单值的读写；一些放松了\*\*\*ACID\*\*\*方法的限制；还有一些不需要预先认定的结构（***schema***）。MySQL用户可以通过使用\*\*\*memcached\*\*\* API直接访问一些类型的MySQL表，将NoSQL风格进程的快速与简单和SQL操作的灵活与方便结合起来。针对InnoDB表的\*\*\*memcached\*\*\*接口在MySQL 5.6及更高版本中可能；如需更多细节，参考[第14.2.16节，InnoDB集成\*\*\*memcached\*\*\*][14.02.16]。针对MySQL Cluster的memached接口在MySQL CLuster 7.2版中可用，如果更多细节，请参考[[http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.html]。](http://dev.mysql.com/doc/ndbapi/en/ndbmemcache.html%5D%E3%80%82)

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [memcached](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_memcached), [schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_schema), [SQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sql).

**NOT NULL constraint: 非空约束**

具体指定一列（***column***）不能包含任何空（***NULL***）值的一类约束（***constraint***）。它有助于保护引用一致性（***referential integrity***），数据库服务器可以识别错误遗漏值的数据。它也可以让优化器来预计一个列上索引中实体的数目，而有助于查询优化器中的算法。

参见 [column](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_column), [constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_constraint), [NULL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_null), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [referential integrity](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_referential_integrity).

**NULL: 空**

一个\*\*\*SQL\*\*\*中特殊的值，用来指代数据的缺失。任何算术运算或等于测试都会引起一个空值，反过来会产生一个空的结果。（因此它类似于IEEE浮点原则中的NaN，“不是一个数据（***not a number***）”。）任何诸如AVG（）这样的聚集计算在决定多少行需要做除数时会忽略掉含有空值的行。唯一可以测试空值的SQL语句是IS NULL或IS NOT NULL。

空值在索引操作中发挥作用，因为对于性能来说，一个数据库必须最小化为了保持跟踪缺失值的负载。通常地，空值不存在索引中，因为在一个在索引列上使用标准比较操作查询是不可能匹配到该列上含有空值的行。因为同样的原因，唯一索引不阻止空值；那些值根本不在索引中。在一个列上定义一个非空约束提供了一个没有行从索引中漏掉的保证，可以更好地查询优化（行的精确行数和评估是否使用索引）。

因为主键（***primary key***）必须能唯一地标识表中的每一行，一个单列主键不能包含任何空值，多列主键在所有列中都不能包含空值。

虽然Orcale数据库允许一个空值可以和字符串连结，但是InnoDB会把这样的操作结果视为空。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [SQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sql).

**O**

**off-page column 跨页列**

一个包含变长数据（诸如BLOB和VARCHAR）的列，其中数据太长而不能适用于B树（***B-tree***）页。数据存储在溢出页（***overflow page***）中。InnoDB \*\*\*Barracuda\*\*\*文件格式中的动态（DYNAMIC）行格式对于这种存储来说，会比老的精简（COMPACT）行格式更有效。

参见 [B-tree](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_b_tree), [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [overflow page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_overflow_page).

**OLTP: 在线联机查询**

“在线联机查询（***Online Transaction processing***）”的简写。一个数据库系统，或一个数据库应用，运行很多事务（***transaction***），伴随频繁的读写，通常一次只影响小部的数据。举个例子，一个航线预订系统或一个处理银行存款的应用。考虑到\*\*\*DML\*\*\*（insert/update/delete）效率和查询（***query***）效率的平衡，数据可能会以范式（***normalized***）形式组织。与之相对的是数据仓库（***data warehouse***）。

凭借其行级锁与事务能力，InnoDB是在应用中使用MySQL表的理想存储引擎。

参见 [data warehouse](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_warehouse), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**online: 在线**

一类不会引起宕机. 阻塞或限制操作数据库的操作。通常适用于\*\*\*DDL\*\*\*。缩短限制操作的操作，诸如快速索引创建（***fast index creation***），在MySQL 5.6中已经大量引入了在线DDL操作（***online DDL***）。

在备份环境中，热备（***hot backup***）是一个在线操作，温备（***warm backup***）的一部分是在线操作。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [Fast Index Creation](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_index_creation), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [online DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online_ddl), [warm backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_warm_backup).

**online DDL: 在线DDL**

一个在\*\*\*DDL\*\*\*（主要是ALTER TABLE）期间提高InnoDB表性能. 并发和可用性的特性。更多细节请参考[第14.2.11节，InnoDB和在线DDL]。

具体细节因操作类型的不同而不同。在某些情况下，表可以在ALTER TABLE运行的同时被并发修改。操作可能可以在不做表拷贝的情况下执行，或使用一个特殊优化类型的表拷贝。空间使用量由配置选项[innodb\_online\_alter\_log\_max\_size]控制。

这个特征是MySQL 5.5和MySQL 5.1 InnoDB Plugin中快速索引创建（***Fast Index Creation***）的加强版。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [Fast Index Creation](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_index_creation), [online](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online).

**.OPT file: .OPT文件**

一个多包含数据库配置信息的文件。带有这个后缀名的文件总是包含在MySQL企业备份产品（***MySQL Enterprise Backup***）的mysqlbackup命令（***mysqlbackup command***）生成的备份中。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command).

**optimistic: 乐观的**

一个引导关系型数据库系统低层实现决策的方法。在一个关系型数据库中对性能和并发（***concurrency***）的要求意味着操作必须快速启动或调度。对一致性和参照完整性（***referential integrity***）的要求意味着任何操作都有可能失败：一个事务可能回滚，一个\*\*\*DML\*\*\*操作可能违反约束，一个对锁的请求可能导致死锁，一个网络错误可能导致超时。乐观策略假设大多数请求或尝试都会成功，所以只做相对来说很小的工作来对付失败的情况。当这个假设为真时，数据库做少许不必要的操作；当请求失败了时，必须要额外的工作来清理和撤消变更。

InnoDB为锁（***locking***）和提交等操作采用乐观策略。例如，事务产生的数据变更会在提交（***commit***）发生前就写到数据文件中，使得提交本身非常快，但如果事务回滚时需要做更多的荏来撤消变更。

与乐观策略相对的是悲观（***pessimistic***）策略，其中的系统优化为处理不可靠靠或频繁失败的操作。这种方法在数据库系统中很罕见，因为更多关注选择可靠的硬件. 网络和算法。

参见 [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [pessimistic](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_pessimistic).

**optimizer: 优化器**

MySQL基于相关表（***table***）的特点与数据分布用，来决定为查询（***query***）采用最优索引（***index***）和关联（***join***）顺序的组件。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [join](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_join), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**option: 选项**

一个MySQL的配置参数，既存在配置文件（***option file***）中，又可以通过命令行命令传递。

对于应用到（***InnoDB***）表上的选项，每个选项名都以前缀innodb\_开头。

参见 [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [option file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_option_file).

**option file: 选项文件**

保存MySQL实例配置选项（***option***）的文件，一般来说，在Linux和UNIX上，这个文件叫my.cnf，在Windows上，这个文件叫my.ini。

参见 [configuration file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_configuration_file), [my.cnf], [option](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_option).

**overflow page 溢出页**

单独申请用来存储因太长而不能放置B树（***B-tree***）页（***page***）的变长列（如BLOB和VARCHAR）的磁盘页。这些相关的列也叫页外列（***off-page column***）。

参见 [B-tree](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_b_tree), [off-page column](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_off_page_column), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**<name="P">P**

**page: 页**

一个数据单元，表示InnoDB任何时刻在磁盘（数据文件，***data files***）与内存（***buffer pool***）之间传输的数据量。一个页可以包含一行或多行，取决于每行有多少数据。如果一行不能完整放到单个页中，InnoDB会设置一个指针类型数据结构，这样行的信息能存到一个页中。

让每行中可以放置更多数据的一种方法是使用压缩行格式（***compressed row format***）。对于使用BLOB或大文本段列的表，精简行格式（***compact row format***）可以让这些大列跟行中的其它内容分开来存储，针对那些没有引用到那些列的查询降低I/O负载和内存使用。

当InnoDB为了提高吞吐量而批量读写一组页时，它会一次读写一个簇（***extend***）。

一个MySQL实例中所有的InnoDB磁盘数据结构都使用同样的页大小（***page size***）。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [compact row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compact_row_format), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [extent](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_extent), [page size](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page_size), [row](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row).

**page cleaner 页清理器（页清理线程）**

一个InnoDB后台线程（***thread***），用来从\*\*\*buffer pool\*\*\*中刷新脏页（***dirty page***）。在MySQL 5.6之前的版本中，该行为由主线程（***master thread***）执行。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [master thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_master_thread), [thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_thread).

**page size: 页大小**

对于发布到MySQL 5.5版本，包括MySQL 5.5，每个InnoDB页（***page***）固定为16KB。这个值代表一个平衡：对于保存绝大多数行来说足够大，对减小加载不必要的数据到内存中的性能开销来说也足够小。其它值未经测试或不支持。

从MySQL 5.6开始，InnoDB实例（***instance***）的页大小可以是4KB. 8KB或16KB，由[innodb\_page\_size]配置选项控制。你在创建MySQL实例的时候设置这个值，之后它会一直保持不变。相同的页大小会应用到所有的InnoDB表空间（***tablespace***）中，无论是系统表空间（***system tablespace***）还是在\*\*\*file-per-table\*\*\*模式下创建的任何独立表空间。

更小的页大小有提升使用比较小的块大小的存储设备，特别是对磁盘受限系统（***disk-bound***）中的\*\*\*SSD\*\*\*设备来说，比如\*\*\*OLTP\*\*\*应用。当单独的行被更新时，更少的数据被拷贝到内存中. 写到磁盘. 重组及锁定等等。

参见 [disbound], [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [instance](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_instance), [OLTP](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_oltp), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [SSD](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ssd), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**.PAR file: .PAR文件**

分区定义表。此后缀命的文件常常包含在由MySQL企业备份（***MySQL Enterprise Backup***）中mysqlbackup命令（***mysqlbackup command***）生产的备份中。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command).

**parent table: 父表**

外键（***foreign key***）关系中保存从子表（***child table***）中指向初始列的值的表。在父表中删除或更新行的结果依赖于外键定义中的ON UPDATE和ON DELETE子句。子表中符合条件的值会被自动依次删除或更新，或这些列被设置为NULL，或操作被拒绝。

参见 [child table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_child_table), [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key).

**partial backup: 部分备份**

一个包含MySQL数据库中部分表（***table***）. 或包含MySQL实例中部分数据库的备份。与之相对的是全备份（***full backup***）。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [full backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_backup), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**ppartial index: 部分索引**

只一部分表示列值的索引，一般来说是长VARCHAR值的前N个字符（前缀，***prefix***）。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [index prefix](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index_prefix).

**Performace Schema: 性能库**

在MySQL 5.5及以上版本中，性能库（***performance\_schema***）提供一组表，你通过它们可以查询到很多MySQL服务内部性能特性。

参见 [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [rw-lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rw_lock).

**persistent statistic: 持久统计**

MySQL 5.6的特性，将InnoDB表（***table***）的索引（***index***）统计存储到磁盘上，为查询（***query***）提供更好的执行计划稳定性（***plan stability***）。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [optimizer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_optimizer), [plan stability](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_plan_stability), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**pessimistic: 悲观**

一种牺牲性能或并发来获得安全的方法。它适用等请求或尝试可能失败的占比很高或失败请求的后果很严重的情况。InnoDB使用悲观锁（***locking***）机制来最小化死锁（***deadlock***）的机会。在应用层，你可以在通过在一开始就申请事务所需要的所有锁的悲观策略来避免死锁。

很多内置的数据库机制使用与它相对的乐观（***optimistic***）方法。

参见 [deadlock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_deadlock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [optimistic](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_optimistic).

**phantom: 幻（读）**

一个出现在查询结果集中，却没有在之前该查询结果集中出现的行。例如，如果一个查询在一个事务（***transaction***）中运行了两次，期间，另一个事务在新插入一行或更新一行后提交了所以在这个查询的WHERE子句中匹配到了。

这种情况叫幻读。这要比非重复读（***non-repeatable read***）要更难防御一些，因为锁住第一次查询结果集中的所有行并不能阻止导致幻读出现的变更。

在不同的隔离级别（***isolation level***）中，幻读在可序列化读（***serializable read***）级别是被阻止的，但在可重复读（***repeatable read***）. 一致性读（***consistent read***）和未提交读（***read uncommitted***）级别是允许的。

参见 [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [non-repeatable read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_non_repeatable_read), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**physical: 物理**

涉及硬件相关方面的一类操作，如磁盘块. 内存页. 文件. 位及读盘等等。一般情况下，物理层面在专家级性能调优和问题诊断中是很重要的。与之相对的是逻辑（***logical***）。

参见 [logical](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_logical), [physical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical_backup).

**physical backup: 物理备份**

一个拷贝实际数据文件的备份（***backup***）。例如，MySQL企业备份（***MySQL Enterprise Backup***）产品的mysqldump命令生成的就是物理备份，因为它的输出包含可以被mysqld直接使用的数据文件，这让恢复（***restore***）操作变得更快。与之相对的是逻辑备份（***logical backup***）。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [logical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_logical_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [restore](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_restore).

**PITR: 定点恢复**

定点恢复（***point-in-time recovery***）的简写。

参见 [point-in-time recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_point_in_time_recovery).

**plan stability: 执行计划稳定性**

一个查询执行计划（***query execution plan***）的属性，其中优化器对于给定的查询（***query***）总是做出相同的选择，这样性能就是不变的和可预知的了。

参见 [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [query execution plan](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query_execution_plan).

**plugin: 插件**

在MySQL 5.1及更早的版本中，一个包含了对InnoDB存储引擎的特性与性能改善的独立可安装形式，这些改善在那些发行版中的内置InnoDB中并未包括。

在MySQL 5.5及更高版本中，MySQL分发版包含最新的InnoDB特性与性能的改善，即InnoDB 1.1，它不再是一个单独的InnoDB插件了。

差别主要在MySQL 5.1中，其中一个特性或bug修复可能要应用到InnoDB插件中，但不是内置的InnoDB中，反之亦然。

参见 [built-in](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_built_in), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb).

**point-in-time recovery: 定点恢复**

通过恢复备份（***backup***）将数据库的状态重建到一个指定的日期与时间点的进程。通常简写为\*\*\*PITR\*\*\*。因为指定的时间点不太可能恰好就是备份的时间点，所以这项技术常常需要一个物理备份（***physical backup***）和一个逻辑备份（***logical backup***）的组合。例如，利用MySQL企业备份（***MySQL Enterprise Backup***）产品，你恢复了指定时间点前的最近一个备份，然后重放备份时间点与定点恢复时间点之间的二进制日志（***binary log***）中的变更。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [logical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_logical_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [physical backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_physical_backup), [PITR].

**prefix: 前缀**

参见 [index prefix](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index_prefix).

**prepared backup: 一致备份**

在所有二进制日志（***binary log***）与增量备份（***incremental backup***）应用阶段都完成后的一组备份文件，由MySQL企业备份（***MySQL Enterprise Backup***）产品生成。结果文件已为恢复（***restore***）做好准备。在应用步骤之前，这些文件叫原始备份（***raw backup***）。

参见 [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [incremental backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_incremental_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [raw backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_raw_backup), [restore](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_restore).

**primary key: 主键**

可以在表中唯一确定每一行的一组列（注：基于这组列的索引）。因此，它必须是一个不含NULL值的唯一索引。

InnoDB要求每张表都有这样一个索引（也叫簇索引，\*\*\*clustered index\*\*\*或\*\*\*cluster index\*\*\*），并且基于主键的列值组织表存储。

当选择一个主键值时，考虑使用任意值（人造键，***synthetic key***），而依靠其它来源得出的值（自然键，***natural key***）。

参见 [clustered index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clustered_index), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [natural key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_natural_key), [synthetic key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_synthetic_key).

**process: 进程**

一个正在执行的程序的实例。操作系统在多个运行的进程之间切换，允许一定级别的并发（***concurrency***）。在大多数据操作系统上，进程可以包含多个执行共享资源的线程（***thread***）。线程之间的上下文切换要快于进程之间的等效切换。

参见[concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_thread).

**pseudo-record: 伪记录**

一个索引中人造的记录，用来锁定（***locking***）当前不存在的键值或范围。

参见 [infimum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_infimum_record), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [supremum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_supremum_record).

**Pthread: 不译**

POSIX线程标准，定义了UNIX和Linux系统上线程和锁操作API。在UNIX和Linux系统上，InnoDB使用为互斥（***mutex***）的使用它的实现。

参见 [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex).

**purge: 清除**

由独立线程执行的一类垃圾回收，按周期表运行。清除包含这些动作：从索引中删去过期的值；物理删除被之前的DELETE语句标记为删除的行。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [delete](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_delete), [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer).

**purge buffering: 清除缓冲**

在将在DELETE操作中的索引变更存储到插入缓冲（***insert buffer***）中而不是直接写盘的技术，这样物理写可以最小化随机I/O。（因为删除是两步操作，所以这个操作缓冲了正常清除之前标记为删除的索引记录的写操作。）它是变更缓冲（***change buffering***）的一种类型；其它类型为插入缓冲（***insert buffering***）和删除缓冲（***delete buffering***）。

参见 [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [change buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffering), [delete buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_delete_buffering), [insert buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffer), insert buffering.

**purge lag: 清除链表**

InnoDB清除链表（***history list***）的另一种叫法。与[innodb\_max\_purge\_lag]配置选项有关。

参见 [history list](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_history_list), [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge).

**purge thread: 清除线程**

InnoDB进程中一个专门用来定期执行清除操作的线程。在MySQL 5.6及更高版本中，多清除线程可以用innodb\_purge\_thread配置选项来使其生效。

参见 [purge](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_purge), [thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_thread).

**Q**

**query: 查询**

在\*\*\*SQL\*\*\*中，一个从一张或多张表（***table***）中读取信息的操作。根据数据的组织和查询的参数不同，查找可能会通过查询（***index***）索引而优化。如果涉及多表，这个查询就叫做关联（***join***）。

因为历史原因，有时“查询”在更广义上被用作内部进程讨论的语句上，包括其它类型的MySQL语句，读如\*\*\*DDL\*\*\*和\*\*\*DML\*\*\*语句。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [join](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_join), [SQL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_sql), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**query execution plan: 查询执行计划**

优化器关于如何最有效执行一个查询（***query***）的决策集，包括使用哪个或哪些索引（***index***），以及其中表的关联（***join***）顺序。执行计划稳定性（***plan stability***）使得对于一个给定的查询，一直会选择相同的计划。

参见 [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [join](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_join), [plan stability](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_plan_stability), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query).

**query log 查询日志**

参见 [general query log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_general_query_log).

**quiesce: 系统静默状态**

为了减少数据库的活动量，常常为诸如ALTER TABLE. 备份（***backup***）或关机（***shutdown***）而准备。有可能会引起尽可能多刷新（***flush***），也有可能不会，因此InnoDB不能继续做后台I/O操作。

在MySQL 5.6及更高版本中，语法FLUSH TABLES ... FOR EXPORT为InnoDB表往磁盘写一些数据，确保易于通过拷贝数据来备份那些表。

参见 [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shutdown).

**R**

**RAID: 磁盘阵列**

“廉价冗余磁盘阵列（Redundant Array of Inexpensive Drives）”的首字母缩写。通过在多个磁盘上分散I/O操作在硬件层获得更大的并发，并且改善低层写操作，否则它们将以顺序执行。

参见 [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency).

**random dive: 随机取样**

一种用来快速估计一列中不同值个数（列的基数，***cardinality***）的技术。InnoDB从索引中随机取样页，并且使用这些数据来估算不同值的个数据。这个操作在每张表首词打开时发生。

起初，取样的页数固定为8；现在，此值取决于设置[innodb\_stats\_sample\_pages]参数。

如何随机取页的方法取决于设置innodb\_use\_legacy\_cardinality\_algorithm参数。默认设置（OFF）比老版本拥有更优的随机性。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality).

**raw backup: 原始备份**

在二进制日志（***binary log***）与任何增量备份（***incremental backup***）中的变更被应用之前，MySQL企业备份（***MySQL Enterprise Backup***）产品生成的初始的备份文件集。在这个阶段，文件尚未为恢复（***restore***）做好准备。当这些变更被应用后，文件就叫一致性备份（***prepared backup***）。

参见 [binary log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_binary_log), [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [ibbackup\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibbackup_logfile), [incremental backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_incremental_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [prepared backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_prepared_backup), [restore](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_restore).

**READ COMMITTED: 隔离级别，不译**

为了性能，使用放宽事务（***transaction***）之间部分保护的锁（***locking***）策略的一种隔离级别（***isolation level***）。事务不能看到其它事务未提交的数据，但他们可以看到当前事务之后启动的另一个事务所提交的数据。所以，事务从来看不到任何错误数据，但数据能不能看到，一定程度上取决于其它事务的时间。

在这个隔离级别下，当一个事务执行UPDATE ... WHERE或DELETE ... WHERE操作，其它事务可能不得不等待。事务可以执行SELECT ... FOR UPDATE和不会造成其它事务等待的LOCK IN SHARE MODE操作。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**READ UNCOMMITTED: 隔离级别，不译**

提供事务之间最小量保护的隔离级别（***isolation level***）。查询使用的锁（***locking***）策略使它们能够在通常会等待另一个事务的情况下继续。尽管如此，额外的性能是用不可靠结果的代价换来的，包括其它其它更正了但尚未提交的数据（被称为脏读，***dirty read***）。使用这个隔离级别要格外小心，要注意结果集可能会不一致或不能重现，取决于同一时刻其它事务在做什么。一般来说，这个隔离级别下的事务只做查询，没有插入. 更新或删除操作。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [dirty read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**read view: 读视图**

InnoDB的\*\*\*MVCC\*\*\*机制所使用的一个内部快照。取决于其隔离级别（***isolation***），某些事务（***transaction***）可以看到那些在事务（某些情况下是语句）启动时刻可以看到的数据值。使用读视图的隔离级别有\*\*\*REPEATABLE READ\*\*\*. \*\*\*READ COMMITTED\*\*\*和\*\*\*READ UNCOMMITTED\*\*\*。

参见 [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [MVCC](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mvcc), [READ COMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_committed), [READ UNCOMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_uncommitted), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**read-ahead: 预读**

异步提前获取一组页（***pages***）（一整个簇，***extend***）到\*\*\*buffer pool\*\*\*中的一种I/O请求，在预期中这些页很快就被用到。线性预读技术基于上一个区中对页访问的模式，提前获取区中的所有页，并且这是自MySQL 5.1 InnoDB Plugin始的所有MySQL版本的一部分。随机预读技术在同一个区中有一定数量的页存在于buffer pool的情况下，将提前获取该区中的所有页。随机预读不是MySQL 5.5的一部分，但在MySQL 5.6中由配置选项[innodb\_random\_read\_ahead]重新引入。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [extent](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_extent), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page).

**read-only transaction: 只读事务**

一类事务，可以通过排除一些涉及为每个事务（***transaction***）创建一个读视图（***read view***）的簿记来达到为InnoDB表的优化。只能执行无锁读（***non-locking read***）的查询。它可以明确地用使用语法sTART TRANSACTION READ ONLY来启动，或在某些条件下自动完成。更多细节参考[第14.2.12.2.3，优化只读事务][14.02.12.02.03]。

参考 [non-locking read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_non_locking_read), [read view](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_view), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**record lock: 索引记录锁**

一个索引记录上的锁（***lock***）。例如，SELECT C1 FROM UPDATE FROM T WHERE C1=10; 阻止其它事务的插入. 更新或删除那些tc.1是10的行。与之对应的是间隙锁（***gap lock***）和行记录锁（***next-key lock***）。

参见 [gap lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap_lock), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [next-key lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_next_key_lock).

**redo: 重做，不译**

当\*\*\*DML\*\*\*语句对InnoDB表产生变更时记录在redo日志（***redo log***）中的以记录为单位的数据。它在崩溃恢复（***crash recovery***）中用于校正于不完整的事务（***transaction***）写入的数据。单调递增的\*\*\*LSN\*\*\*值代表通过redo日志的redo数据的积累量。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [LSN](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lsn), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**redo log: 重做日志，不译**

崩溃恢复（***crash recovery***）过程中使用一种基于磁盘的数据结构，用于校正由不完整事务（***transaction***）写入的数据。在正常操作过程中，它将变更InnoDB表数据的请求编码，这些请求由SQL语句或低层的API调用NoSQL接口产生。在异常关机（***shutdown***）之前完有完成更新数据文件（***data files***）的修改会被自动重放。

redo log在物理上表现为一组文件，通常命名为ib\_logfile0和iblogfile1。redo日志中的数据根据受影响的记录编码；这些数据统称为\*\*\*redo\*\*\*。redo日志中通过的数据表现为单调递增的\*\*\*LSN\*\*\*值。在MySQL 5.6.3中，之前redo日志大小4GB的上限升至512G。

redo log在磁盘上的布局取决于配置选项[innodb\_log\_file\_size]. [innodb\_log\_group\_home\_dir]及[innodb\_log\_files\_in\_group](https://github.com/xufubo/manual56/blob/master/docs/%E5%9F%BA%E6%9C%AC%E4%B8%8D%E7%94%A8)。redo日志操作的性能也受日志缓冲的影响，由配置选项[innodb\_log\_buffer\_size]控制。

参见 [crash recovery](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_crash_recovery), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [ib\_logfile](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ib_logfile), [log buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log_buffer), [LSN](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lsn), [redo](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo), [shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shutdown), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**redundant row format: 冗余行格式**

最早的InnoDB行格式，对于使用Antelope文件格式的表有效。在MySQL 5.0.3之前，它是InnoDB中唯一有效的行格式。在MySQL 5.0.3及更高版本中，默认为精简行格式。你仍然可以指定冗余行格式以兼容较老的InnoDB表。

如需更多关于InnoDB冗余行格式的信息，请参考[第14.2.9.4节，精简与冗余行格式][14.02.09.04]。

参见 [Antelope](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_antelope), [compact row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compact_row_format), [file format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_format), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**referential integrity: 参照完整性**

维护数据一直处于一致格式的技术，\*\*\*ACID\*\*\*理念的一部分。实际上，不同表中的数据通过使用外键约束（***FOREIGN KEY constraint***）来保持一致，它能阻止变更发生或自动像变更传递到关联的表中。关联机制包括阻止即将误插入重复值的唯一约束（***unique constraint***）和阻止即将误插入空白值的非空约束（***NOT NULL constraint***）。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [FOREIGN KEY constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key_constraint), [NOT NULL constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_not_null_constraint), [unique constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_constraint).

**relational: 关系**

现代数据库系统很重要的一部分。数据库服务编码并强制实施关系，如一对一. 一对多和唯一性。例如，一个人在一个地址本数据库中可能拥有零个. 一个或多个手机号；单个电话号码可能与多个家庭成员关联。在财务数据库中，一个人可能要求刚好拥有一个纳税人ID，并且任何纳税人ID只能与一个人关联。

数据库服务可以利用这些关系来阻止错误数据被插入，并且找到高效查询信息的方法。例如，如果一个值被定义为唯一，那么服务可以在找到第一个匹配项之后就停止搜索，并且他可以拒绝试图插入相同值的第二份拷贝。

在数据库层面，这些关系通过SQL特性来表示，诸如表中的列（***column***）. 唯一约束. 非空约束（***NOT NULL constarnts***）. 外键（***foreign key***）及不同类型的关联操作等。复杂的关系一般会引用数据分拆到多个表中。通常，数据是范式化（***normalized***）的，所以一对多关系中的重复的值只被存储一次。

在数学语境中，数据库中的关系起源于集合论。比如，WHERE子句中的OR和AND操作表现为并集和交集的概念。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_constraint), [foreign key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_foreign_key), [normalized](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_normalized).

**relevance: 相关度**

在全文搜索（***full-text search***）特性中，一个表示搜索字符串与全文索引（***FULLTEXT index***）中数据相似度的数字。例如，当你搜索一个单词时，该词出现多次的行相比只出现一次的行，单词与它的相关性要更大一些。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**REPEATABLE READ: 隔离级别，不译**

InnoDB默认的隔离级别（***isolation level***）。它阻止查询任何正在被其它事务修改的行，从而阻断了非重复读（***non-repeatable read***）但不能阻断幻（***phantom***）读。它使用了一个适度严格的锁（***locking***）策略，所以所有在同一个事务中的查询能从同一个快照上读到数据，也就是说，数据跟事务启动时一样的。

当一个事务在这个隔离级别下执行UPDATE ... WHERE. DELETE ... WHERE. SELECT ... FOR UPDATE和LOCK IN SHARE MODE操作时，其它事务可能需要等待。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [phantom](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_phantom), [SERIALIZABLE](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_serializable), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**replication: 复制**

从主库（***master database***）发送变更到一个或多个从库（***slave database***）的做法，这样所有的数据库拥有相同的数据。这个技术有着广泛的应用，如为更好的扩展性而做的负载均衡. 容备以及测试软件升级与配置变更。变更在数据库之间发送的方法有基于行的复制（***row-based replication***）与基于语句的复制（***statement-based replication***）。

参见 [row-based replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_based_replication), [statement-based replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_statement_based_replication).

**restore: 恢复**

将一组文件从MySQL企业备份产品（***MySQL Enterprise Backup***）中放置到MySQL中的进程。执行这个操作可以修改损坏了的数据库. 返回到早期某个时间点或设置一个新的从库（在复制（***replication***）环境下）。在MySQL企业备份产品（***MySQL Enterprise Backup***）中，这个操作由mysqlbackup命令（***mysqlbackup command***）的copy-back选项来执行。

参见 [hot backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_hot_backup), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command), [prepared backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_prepared_backup), [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication).

**rollback: 回滚**

一个结束一个事务（***transaction***），撤消任何由该事务生成的变更的\*\*\*SQL\*\*\*语句。它与提交相对（***commit***），提交是让该事务产生的变更保持持久。

默认情况下，MySQL使用自动提交（***autocommit***）设置，它在每条SQL语句之后自动执行提交。如果你要使用回滚技术的话，必须要变更这个设置。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**rollback segment: 回滚段**

包含undo日志（***undo log***）的存储区，是系统表空间（***system tablespace***）的一部分。

参见 [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**row: 行**

有一组列（***column***）定义的逻辑数据结构。一组行组成一张表（***table***），在InnoDB数据文件（***data files***）中，每个页可以包含一行或多行。

尽管InnoDB为了与MySQL语法保持一致而使用了术语行格式（***row format***），但行格式是每张表的一个属性并且应用到了表中的所有行上。

参见 [column](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_column), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_page), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**row format: 行格式**

InnoDB表（***table***）中行（***row***）的磁盘存储格式。当InnoDB有了如压缩等新的功能时，新的行格式会被引进，以支持存储在效率与性能方面的改善。

每一张表都有自己的行格式，通过ROW\_FORMAT选项来指定。要查看每张InnoDB表的行格式，执行命令SHOW TABLE STATUS。因为系统表空间中的所有表使用同一种行格式，所以要利用其它行格式的优势的话，一般要求打开[innodb\_file\_per\_table]选项，以便每张表都存储在独立的表空间中。

参见 [compact row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compact_row_format), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [dynamic row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dynamic_row_format), [fixed row format], [redundant row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redundant_row_format), [row](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**row lock: 行锁**

一种锁（***lock***），阻止一行被别一个事务（***transaction***）以互斥的方式访问。同一个表中的其它行则可以被其它事务自由的写入。这是在\*\*\*InnoDB\*\*\*表上做\*\*\*DML\*\*\*操作产生的一类锁（***locking***）。

与之对应的是MyISAM对应的表级锁（***table lock***），或\*\*\*DDL\*\*\*操作期间的用在不能做在线DDL（***online DDL***）操作的InnoDB表上的锁；这些锁阻止对表的并发访问。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [online DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online_ddl), [table lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**row-based replication: 行复制**

一种复制（***replication***）的形式，其中事件从主库（***master***）传播到从库（***slave***）上，并说明独立的行如何发生变更。它对于[innodb\_autoinc\_lock\_mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb_autoinc_lock_mode)选项的所有设置都是安全的。

参见 [auto-increment locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_auto_increment_locking), [innodb\_autoinc\_lock\_mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb_autoinc_lock_mode), [master server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_master_server), [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication), [slave server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slave_server), [statement-based replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_statement_based_replication).

**row-level locking: 行级锁**

\*\*\*InnoDB\*\*\*表使用的锁（***locking***）机制，依靠行锁（***row lock***）而非表锁（***table lock***）。多个事务（***transaction***）可以同时修改同一张表。只有当两个事务试图修改同一行数据时让其中一个事务等待其它事务完成（并且释放它的行锁）。

参见 [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_lock), [table lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**rw-lock: 读写锁**

InnoDB低级对象，用来表示和强制对内部内存结构体共享访问锁（***lock***）。一旦这个锁被获取了，任何其它的进程和线程等等能读到数据结构，但是没有一个可以写它。与之对应的是互斥锁（***mutex***），它强制排它访问。互斥锁和读写锁统称为闩锁（***latch***）。

参见 [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema).

**savepoint: 保存点**

保存点有助于实现嵌套事务（***transaction***）。它们常用来提供在表上操作的范围，该表是大事务的一部分。例如，在预订系统中安排一个旅行可能会涉及到预订几个不同的航班；如果一个中意的航班不可用，你可以只回滚（***rollback***）预订中涉及单程的变更，而不必将之前成功预约的航班都回滚。

参见 [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**scalability: 可扩展性**

往一个系统中增加更多工作，以及处理更多并发请求的能力，并在突破系统容量上限期间并无性能的突然下降。软件架构. 硬件配置. 应用编码以及负载类型都在可扩展性中扮演重要角色。当系统到达最大容量，提高可扩展性的流行的做法是垂直扩展（***scale up***，提升已有软硬件的能力）和水平扩展（***scale out***，增加新的服务器和更多MySQL实例）。常常与可用性（***availability***）搭配作为一个大规模部署中的关键组成部分。

参见 [availability](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_availability), [scale out](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scale_out), [scale up](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scale_up).

**scale out: 水平扩展**

通过增加新服务器和更多MySQL实例来提高可扩展性（***scalability***）的技术。例如，架设复制. MySQL集群. 连接池或其它通过一组机器来扩展工作能力的方面。与之对应的是垂直扩展（***scale up***）。

参见 [scalability](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scalability), [scale up](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scale_up).

**scale up: 垂直扩展**

通过提升已有软硬件能力来提高可扩展性（***scalability***）的技术。例如，给一台服务器增加内存并调整与内存有关的参数，如[innodb\_buffer\_pool\_size]和[innodb\_buffer\_pool\_instances]。与之对应的是水平扩展（***scale out***）。

参见 [scalability](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scalability), [scale out](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_scale_out).

**schema: 指数据库（仅MySQL）**

从概念上讲，schema指的是一组相互之间有关联的数据库对象，如表、表列、列的数据类型、索引、外键等等。这些对象通过SQL语法联系在一起，因为列组成表，外键指向表和列等。**理想情况下**，它们在逻辑上也是连在一起的，做为统一的应用或灵活的框架的一部分来一起工作。例如，information\_schema和performance\_schema数据库在它们的名字中使用"schema"来强化它们所包含表与列之间的亲近关系。在MySQL中，从物理上讲，一个schema等同于一个数据库。你可以在MySQL语法中，将关键字DATABASE用SCHEMA替代，例如，用CREATE SCHEMA来替代CREATE DATABASE。

一些其它的数据库产品会有区别。例如：在Oracle数据库产品中，schema只代表数据库的一部分：属于一个单个用户的表和其它对象。

**search index: 搜索索引**

在MySQL中，全文搜索（***full-text search***）查询使用一种特殊的索引，全文索引（***FULLTEXT index***）。在MySQL 5.6.4及以上版本中，InnoDB和MyISAM表都支持全文索引了；在这之前，这些索引只在MyISAM表中可用。

参见 [full-text search](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_full_text_search), [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**secondary index: 二级索引**

相当于表列的子集的一类InnoDB索引（***index***）。一个InnoDB表可以拥有0个. 一个或多个二级索引。（与之对应的是聚集索引，***clustered index***，每个InnoDB都要用有这个索引，并且存储了表中所有的列。）

一个二级索引可以用来满足只从索引列中请求数据的查询。对于更加复杂的查询，它可以用来找到表中有关的行，这些行之后会通过查询聚集索引来获取。

创建和删除二级索引通常会因为拷贝所有InnoDB表的数据而带来的明显的性能负载。InnoDB Plugin的快速索引创建（***fast index creation***）特性会加速针对二级索引的CREATE INDEX和DROP INDEX语句。

参见 [clustered index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clustered_index), [Fast Index Creation](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_index_creation), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index).

**segment: 段**

InnoDB表空间（***tablespace***）中主要组织结构。如果表空间像是一个目录的话，段就像是目录里的文件。段会增长，新段会被创建。

例如，在\*\*\*file-per-table\*\*\*的表空间中，表数据在一个段中，每个相关的索引在自己的段中。系统表空间（***system tablespace***）包含多个不同的段，因为它要保存多个表和它们相关的索引。系统表空间也包括最多由128个回滚段（***rollback segments***）组成的undo日志（***undo log***）。

段会随着数据的插入与删除会增大与缩小。当一个段需要更多空间，它会一次扩展一个簇（***extent***，1MB）。类似地，当某个簇中的数据不再需要时，段会释放掉这个簇中可利用的空间。

参见 [extent](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_extent), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [rollback segment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback_segment), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**selectivity: 选择性**

数据分布的一个属性，一个列中不同值的数（基数，***cardinality***）除以表中的记录数。高选择性意味着列中的值相对唯一，且可以高效地通过索引来获取数据。在你（或查询优化器）可以预测到一个尝试在WHERE子句只能匹配少量（或部分）表中的行的情况下，如果首先使用索引评估这个尝试，整个查询（***query***）往往是高效的。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query).

**semi-consistent read: 半一致性读**

UPDATE语句使用的一类读操作，是提交读（***read committed***）和一致性读（***consistent read***）的组合体。当一个UPDATE语句检查一个已经被锁了的行时，InnoDB返回给MySQL最近提交的版本以便MySQL可以判断此行是否满足UPDATE的WHERE条件。如果行匹配（必须要被更新），MySQL重读该行，并且这次InnoDB会锁住它或等待为它上锁。这类读操作只有在事务在读提交隔离级别（***isolation level***）时发生，或在[innodb\_locks\_unsafe\_for\_binlog]选项开启时发生。

参见 [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [READ COMMITTED](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_read_committed).

**SERIALIZABLE: 隔离级别，不译**

使用最保守锁机制的隔离级别（***isolation level***），阻止任何其它的事务插入或变更本事务读取的数据，直到它结束。这种情况下，同一个查询可以在一个事务里一遍又一遍地运行，并且每次都能确保读到相同的结果集。任何对被当前事务启动之后的事务所提交数据的变更尝试，都会导致当前事务的等待。

这是SQL标准指定的默认隔离级别。实际上，这么严格的级别是很少需要的，所以InnoDB的默认隔离级别是次严格的，可重复读（***repeatable read***）。

参迎 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [REPEATABLE READ](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_repeatable_read), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**server: 服务（器）**

持续运行的一程序，等待接收并处理另一个程序（客户端，***client***）发来的请求。因为一整台计算机常常专门用于运行一个或多个服务（数据数据库服务. web服务. 应用服务或它们的组合），所以这个术语\*\*\*server\*\*\*也用来指运行这个服务软件的计算机。

参见 [client](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_client), [mysqld](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqld).

**shared lock: 共享锁**

一种锁（***lock***），允许其它事务（***transaction***）读取锁定了的对象，也可以在它之上再加共享锁，但不能写它。与它相反的是排它锁（***exclusive lock***）。

参见 [exclusive lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_exclusive_lock), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**shared tablespace: 共享表空间**

同系统表空间（***system tablespace***）。

参见 [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace)。

**sharp checkpoint: 清晰检查点**

将所有buffer pool中的redo实体包含在redo日志（***redo log***）的某个地方的脏页（***dirty page***）全部刷新（***flush***）到磁盘的进程。发生在InnoDB重用一个log文件的一部分之前；log日志沿环性使用。通常发生在写敏感的工作负载下（***workload***）。

参见 [dirty page](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dirty_page), [flush](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_flush), [redo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redo_log), [workload](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_workload).

**shutdown: 关闭**

停止MySQL服务的进程。默认情况下，这个进程对InnoDB表做清扫操作，所以它可以缓慢关闭，但是之后会很快启动。如果你略过了清扫操作，它很快关闭但在下次重启时必须做清扫工作。

快速关闭模式由[innodb\_fast\_shutdown]选项控制。

参见 [fast shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_shutdown), [InnoDB](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb), [slow shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_slow_shutdown), [startup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_startup).

**slave server: 从服务器**

常常简写为“slave”。复制（***replication***）环境下的一个数据库服务器（***server***），从另一台服务器上接收变更并应用那些同样的变更。所以它维护着与主库（***master***）一样的数据，虽然它有一些延后。

在MySQL，从服务器常常通过替换崩溃了的主服务器来做灾难恢复。它们也常常用来做软件升级和新设置的测试，来确保数据库的配置变更不会给性能或可靠性带来问题。

从服务器通常有高的负载，因为它要处理所有的从主库同步过来的\*\*\*DML\*\*\*（写）操作和用户的查询。为了确保从服务器可以足够快速地应用从主库而来的变更，它们常常使用快速的I/O设备和足够的CPU和内存来运行同一台从服务器上的多个数据库实例。例如，主服务器可能使用机械硬盘，而从服务器使用固态硬盘（***SSD***）。

参见 [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication), [server](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_server), [SSD](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ssd).

**slow query log: 慢查询日志**

一类用来针对MySQL服务处理过的SQL语句进行性能调优的日志（***log***）。日志信息存储在一个文件中。你可以启用这个特性来使用它。你控制记录哪种类型的“慢”SQL语句。如需更多信息，参考[第5.2.5节，慢查询日志][05.02.05]。

参见 [general query log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_general_query_log), [log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_log).

**slow shutdown: 慢关闭**

在关闭操作之前还要做额外的InnoDB刷新操作的一类关闭（***shutdown***）。也叫\*\*\*clean shutdown\*\*\*。由配置参数innodb\_fast\_shutdown=0或命令SET GLOBAL innodb\_fast\_shutdown=0来指定；虽然关闭本身要花更长时间，但它却节省下来的是下次启动的时间。

参见 [clean shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clean_shutdown), [fast shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_shutdown), [shutdown](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_shutdown).

**snapshot: 快照**

数据在一个具体时间点的表示，即使数据被其它事务（***transaction***）的变更提交（***commit***）了，它也保持着相同的数据。在某些隔离级别（***isolation level***）中用来做一致性读（***consitent read***）。

参见 [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**space ID: 表空间ID**

MySQL实例中用来标志InnoDB表空间（***tablespace***）唯一标识。对于系统表空间（***system tablespace***）来说，表空间ID一直为0；系统表空间中所有的表都使用同一个ID。在[file-pre-table]模式下创建的每个表空间文件也拥有自己的空间ID。

在MySQL 5.6之前，硬编码的值使得在MySQL实例之间移动表空间变得困难重重。自MySQL 5.6始，你可以使用包含FLUSH TABLES FOR EXPORT，ALTER TABLE ... DISCARD和ALTER TABLE ... IMPORT TABLESPACE语句的可传输表空间（***transportable tablespace***）特性，在实例间拷贝表空间文件。需要调整的表空间ID的信息通过你与表空间一起拷贝的.cfg文件（\*\*\*.cfg file\*\*\*）传达。如果更多信息，参考[第14.2.5.5节，往另一台服务拷贝表空间（可传输表空间）]。

参见 [.cfg file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cfg_file), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace), [transportable tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transportable_tablespace).

**spin: 自旋**

持续尝试一个资源是否可用的一种等待。这种技术用在那些通常只在短时间内被持有的资源上，这种情况下在一个“繁忙的循环”下等待要比把一个线程推到休眠状态并执行一个上下文切换要高效得多。如果资源在短时间内没有变为有效，自旋循环终止，另一种等待技术上阵。

参考 [latch](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_latch), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [mutex](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mutex), [wait](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_wait).

**SQL: 结构化查询语言**

结构化查询语言是标准执行数据库操作的语言。常常分类为\*\*\*DDL\*\*\*. \*\*\*DML\*\*\*和查询（***query***）。MySQL包含一些附加的语句类型，如复制（***replication***）。SQL语法基础参考[第9章，语言结构][9]，MySQL表的列所使用的数据类型参考[第11章，数据类型][11]，查询中所用到的标准的和MySQL特有的函数参考[第12章，函数与操作符][12]。

参见 [DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ddl), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [replication](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_replication).

**SSD: 固态驱动器**

固态驱动器（solid-state drive）的缩写。与传统硬盘驱动器（***HDD***）有着不同的性能特性的一类存储设备：更小的存储容量、更快的随机写、无移动部件以及大量为写性能所做的考虑。它的性能特性会对磁盘受限（***disk-bound***）的系统产生影响。

参见 [disk-bound](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_disk_bound), [HDD].

**startup: 启动**

启动MySQL服务的过程。

**statement-based replication: 基于语句的复制**

一种复制（***replication***）的形式，将主库（***master server***）中发出并执行的SQL语句，在从库（***slave server***）上重放。为了避免自增锁（***auto-increment locking***）带来潜在时间问题，在使用选项innodb\_autoinc\_lock\_mode时要多加小心。

**statistics: 统计信息**

与每个InnoDB表（***table***）和索引（***index***）有关的评估值，用来构建一个高效的查询执行计划（***query execution plan***）。主要的值是基数（不重复的值数，***cardinality***）和表的行数或索引实体数。表的统计信息体现了它们主键索（***primary key***）引中的数据。二级索引（***secondary key***）的统计信息体现了索引覆盖的行。

因为时时刻刻不同的事务（***transaction***）都可以插入或删除表中的数据，所以这些值是评估的而不是准确的。为了避免这些值被频繁地重算，你可以启用持久统计（***presistent statistics***），其中的值存在InnoDB的系统表中，而且只有当你执行ANALYZE TABLE语句时才会刷新。

你可以通过[innodb\_stats\_method]配置选项来控制在计算统计信息时如何处理NULL值。

针对数据库对象与数据库活动的其它类型的统计信息可以在\*\*\*INFORMATION\_SCHEMA\*\*\*表和\*\*\*PERFORMANCE\_SCHEMA\*\*\*表中找到。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [INFORMATION\_SCHEMA](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_information_schema), [NULL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_null), [Performance Schema](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_performance_schema), [persistent statistics](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_persistent_statistics), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [query execution plan](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query_execution_plan), [secondary index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_secondary_index), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**stemming: 词干**

全文检索相关的东西。

**stopword: 停用词**

在全文索引（***FULLTEXT index***）中，考虑到一个字太普通或太微不足道，所以它从搜索索引（***search index***）中被略过，并且在搜索查询中被忽略。针对不InnoDB表和MyISAM表，有不同的配置选项来控制对停用词的处理。更多细节请参考[第12.9.4节，全文停用词][12.09.04]。

参见 [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index), [search index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_search_index).

**storage engine: 存储引擎**

MySQL数据库的一个组件，用来执行低层的数据存储、更新与查询。在MySQL 5.5及更高版本中，InnoDB取代MyISAM，成为新表的默认存储引擎。不同的存储引擎都是为平衡各方面的因素而设计，比如内存使用与磁盘使用、读写速度、鲁棒性等。每一种存储引擎管理指定的表，所以我们叫InnoDB表、MyISAM表等等。

MySQL企业备份针（***MySQL Enterprise Backup***）对InnoDB表的备份做了优化。它也可以备份MyISAM和其它引擎所处理的表。

**strict mode: 严格模式**

innodb\_strict\_mode选项的常用名。打开这个设置会导致在某些在正常情况下发出警告的场景，会被视为视为错误。例如，某些文件格式（***file format***）与行格式（***row format***）相关的选项的无效组合，正常情况下会产生一个报警，并且用默认值继续，而现在则会导致CREATE TABLE操作失败。

MySQL也有一些叫严格模式的东西。

参见 [file format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_format), [innodb\_strict\_mode](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_innodb_strict_mode), [row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_format).

**sublist: 子列表**

在表示buffer pool的列表结构体中，由不同的列表部分表示比如新的页和比较老的页。一组参数来控制这些部分的大小和新老页的分割点。

参见 [buffer pool](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_buffer_pool), [eviction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_eviction), [list](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_list), [LRU](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lru).

**supremum record: 上确界记录**

一个索引中的伪造（***pseudo-record***）值，用来表示索引中最大值之上的间隙（***gap***）。如果一个事务有一个像SELECT ... FOR UPDATE ... WHERE col >10; 这样的语句，并且 这个列中的最大值是20，就会有一个在上确界记录上的锁来阻止其它事务插入像50、100等更大的值。

参见 [gap](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_gap), [infimum record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_infimum_record), [pseudo-record](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_pseudo_record).

**surrogate key: 代理键**

人造键（***synthetic key***）的别称。

参见 [synthetic key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_synthetic_key).

**synthetic key: 人造键**

一个索引列，通常是主键，其中的值被随意分配。常常由一个自增列来完成。通过完全随意分配该值，你可以避免过于严格的规则和应用错误的假设。比如，如果批准一个员工雇用但从来这个员工实际没有加入过，表示员工号的数据序列可能会产生间隙。再比如，如果有人离开公司，后来又行话加入了，100号的员工的入职时间可能要比500号的员工要晚。数字值也会产生更短的可预计的值。例如，存储代表“马路”、“城市路”、“高速公路”等的代码值要比一遍又一遍地存储这些字串符来说，空间效率要更高一些。

也叫代理键（***surrogate key***）。相对应的是自然键（***natural key***）。

参见 [auto-increment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_auto_increment), [natural key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_natural_key), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [surrogate key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_surrogate_key).

**system tablespace: 系统表空间**

一小集合包含InnoDB相关对象（数据字典，***data dictionary***）以及undo日志（***undo log***）、变更缓冲（***change buffer***）和双写缓冲（***doublewrite buffer***）存储区域的数据文件（即\*\*\*ibdata files\*\*\*）。根据[innodb\_file\_per\_table]的选项，当表被创建时，它也可能包含部分或所有InnoDB表的数据和索引数据。系统表空间中的数据和元数据应用到MySQL实例（***instance***）中的所有数据库（***database***）中。

直到MySQL 5.6.7，默认情况是在系统表空间中保存所有InnoDB表和索引数据，常常导致这些文件变得非常大。因为系统表空间从不收缩，所以如果大量的临时数据被加载然后再被删除，就会出现存储空间问题。在MySQL 5.6.7以及更高版本中，默认就是file\_per\_table模式，其中每个表和它们关联的索引存储在一个独立的.ibd文件（\*\*\*.ibd file\*\*\*）中。新的默认值使得它更易于使用InnoDB依赖\*\*\*Barracuda\*\*\*文件格式的特性，如表压缩与动态（***DYNAMIC***）行格式。

在MySQL 5.6及更高版本中，为[innodb\_undo\_tablespace]选项设置一个值，会将undo日志（***undo log***）分裂到一个或多个独立的表空间文件中。这些文件也被认为是系统表空间的一部分。

保存所有表数据在系统表空间或独立的.ibd文件中一般会对存储管理带来影响。MySQL企业备份产品（***MySQL Enterprise Backup***）可能会备份一小组合的大文件，或很多更小的文件。在有上千表的系统中，文件系统处理上千的.ibd文件会引发瓶颈。

参见 [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [change buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_change_buffer), [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [database](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_database), [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer), [dynamic row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dynamic_row_format), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [ibdata file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibdata_file), [innodb\_file\_per\_table], [instance](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_instance), [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**T**

**.TRG file**

一个包含触发器参数的文件。该后缀名的文件一直包含在由MySQL企业备份（***MySQL Enterprise Backup***）产品的mysqlbackup命令（***mysqlbackup command***）生成的备份文件中。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command), [.TRN file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_trn_file).

**.TRN file**

一个包含触发器命名空间信息的文件。该后缀名的文件一直包含在由MySQL企业备份（***MySQL Enterprise Backup***）产品的mysqlbackup命令（***mysqlbackup command***）生成的备份文件中。

参见 [MySQL Enterprise Backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysql_enterprise_backup), [mysqlbackup command](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_mysqlbackup_command), [.TRG file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_trg_file).

**table: 表**

每一张MySQL表都与一种特定存储引擎（***storage engine***）相关联。\*\*\*InnoDB\*\*\*表有特定的物理（***physical***）与逻辑（***logcial***）特性，这些特性会影响性能、扩展性（***scalability***）、备份（***backup***）、管理和应用部署。

在文件存储术语中，如果表是在\*\*\*file-per-table\*\*\*模式下创建的，每一张InnoDB表既是单个大InnoDB系统表空间（***system tablespace***）的一部分，又是独立的.ibd文件的一部分。.ibd文件保存所有表和它的索引（***index***）的数据，也就是表空间（***tabelspace***）。

在file-per-table模式下创建的InnoDB表可以使用\*\*\*Barracuda文\*\*\*件格式。Barracuda表可以使用动态行格式（***DYNAMIC row format***）或压缩行格式（***COMPRESSED row format***）。这些对应的新设置启用了大量的InnoDB特性，比如压缩（***compression***）、快速索引创建（***fast index creation***）和溢出页列（***off-page columns***）。

为了向后兼容MySQL 5.1及更早版本，系统表空间中的InnoDB表必须使用\*\*\*Antelope\*\*\*文件格式，它支持精简行格式（***compact row format***）与冗余行格式（***redundant row format***）。

InnoDB的行（***row***）由叫聚集索引（***clustered index***）的的索引结构体组织起来，实体基于表的主键（***primary key***）列排序。在主键列上的过滤与排序的查询的数据访问是优化的，并且每个索引中包含每个实体对应的主键拷贝。更新主键列中的任何值都是一个开销很大的操作。因此InnoDB表设计的一个重要因素是将最重要查询的列选为主键，并保持主键短小，键值极少变更。

参见 [Antelope](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_antelope), [backup](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_backup), [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [clustered index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_clustered_index), [compact row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compact_row_format), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [compression](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compression), [dynamic row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dynamic_row_format), [Fast Index Creation](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fast_index_creation), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [off-page column](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_off_page_column), [primary key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_primary_key), [redundant row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_redundant_row_format), [row](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**table lock: 表级锁**

一个阻止其它事务（***transaction***）访问表的锁（***lock***）。InnoDB做了很大的努力，通过在线DDL（***online DDL***）、行锁（***row lock***）和针对处理\*\*\*DML\*\*\*语句和查询（***query***）的一致性读（***consistent read***）等技术，使得这种锁不再需要。你可以通过使用LOCK TABLE的SQL语句来创建一个这样的锁；从其它数据库系统和MySQL存储引擎迁移数据的步骤之一就是删除这样的语句，不管实不实用。

参见 [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [DML](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_dml), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [locking](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_locking), [online DDL](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_online_ddl), [query](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_query), [row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_row_lock), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction).

**tablespace: 表空间**

一个可以保存一个或多个InnoDB表（***table***）和对应索引（***index***）数据的文件。系统表空间（***system tablespace***）包含构成数据字典（***data dictionary***）的表，在MySQL 5.6之前默认保存所有其它的InnoDB表。[innodb\_file\_per\_table]选项在MySQL5.6及更高版本中默认为打开状态，打开这个选项允许所新建的表都拥有息的表空间，每个表一个独立的数据文件（***data file***）。

通过打开[innodb\_file\_per\_table]选项来使用多个表空间，这对于使用像压缩和可传输表空间以及管理磁盘使用是必不可少的。参见[第14.2.5.2节，InnoDB File-Per-Table模式][14.02.05.02]。

由内置的InnoDB存储引擎创建的表空间向上兼容InnoDB Plugin。如果表空间使用\*\*\*Antilope\*\*\*文件格式，由InnoDB Plugin创建的表空间向下兼容内置的InnoDB存储引擎。

MySQL集群也把它的表归组到表空间中，如需更多细节，参见[17.5.12.1节，MySQL集群数据对象][17.05.12.01]。

参见 [Antelope](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_antelope), [Barracuda](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_barracuda), [compressed row format](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_compressed_row_format), [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [data files](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_files), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_index), [innodb\_file\_per\_table], [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**tablespace dictionary: 表空间数据字典**

InnoDB表空间中，一个表数据字典（***data dictionary***）元数据的体现。当表被打开后，该元数据可以通过.frm文件（\*\*\*.frm file\*\*\*）检查一致性，来诊断由过期的.frm文件引起的错误。该信息为在系统表空间（***system tablespace***）中的表而准备，也为因\*\*\*file-per-table\*\*\*选项而拥有它们自己.ibd文件（\*\*\*.ibd file\*\*\*）的表而准备。

参见 [data dictionary](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_data_dictionary), [file-per-table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_file_per_table), [.frm file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_frm_file), [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**temporary table: 临时表**

一张表（***table***），它的数据不需要做执久化。例如，临时表可能被作是复杂计算或变换中间结果的存储区域；这个中间数据可能不需要在崩溃后恢复。通过减少对数据写盘的严谨性与其它在重启时保护数据的措施，数据产品可以采取多种便捷来提高在临时表上操作的性能。

有时，数据自己会在一个设定的时候自动删除，比如在事务结束时或在会话结束时。在一些数据库产品中，表自己也会自动删除。

参见 [table](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_table).

**temporary tablespace: 临时表空间**

针对没有压缩的InnoDB临时表和相关对象的表空间。该表空间在MySQL 5.7.1中引入。配置文件选项innodb\_temp\_data\_file\_path允许用户为临时数据文件定义一个相关的路径。如果innodb\_temp\_data\_file\_path没有指定，默认行为是创建一个单独的自动增长的12MB的数据文件，位置在数据目录下的ibdata1边上，叫ibtmp1。临时表空间在每次服务启动时重建并接收到一个动态生成的空间id，它有助于避开与已有表空间id发生冲突。临时表空间不能放在裸设备上。无法创建临时表或创建错误会当成致命错误并且服务启动被拒绝。

在正常关机或初始化中止时表空间会被删掉，例如，一个用户指定了错误的启动选项时可能产生了中止。临时表空间在崩溃发生时不能被删除。在这种情况下，数据库管理员可以手动删除这个表空间或用同样的配置重启服务，这将会删除或重建临时表空间。

参见 [ibtmp file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibtmp_file).

**text collection: 文本集合**

包含中全文索引（***FULLTEXT index***）中的一个列集。

参见 [FULLTEXT index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_fulltext_index).

**thread: 线程**

处理的一个单元，一般要比进程（***process***）更轻量一些，允许更大的并发（***concurrency***）。

参见 [concurrency](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_concurrency), [master thread](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_master_thread), [process](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_process), [Pthreads](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_pthreads).

**torn page: 残缺页**

一种错误状态，它会因I/O设置配备和硬件错误的组合而发生。如果数据被写入的块要比InnoDB的页大小（***page size***，默认为16K）要小，写入时的硬件错误会导致页的一部分存储到磁盘上。InnoDB的双写缓冲（***doublewrite buffer***）避免这种可能性。

参见 [doublewrite buffer](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_doublewrite_buffer).

**TPS 每秒事务**

每秒事务（***transations*** per second）首字母缩写，有时用在基准测试中的一种测量单位。它的值取决于特定基准测试中所体现的负载（***workload***），结合你所控制的硬件容量与数据配置因素。

参见 [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [workload](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_workload).

**transaction: 事务**

事务是可以提交（***commit***）或回滚（***rollback***）的原子工作单元。当一个事务造成数据库中多处变更时，要么在事务提交时所有的变更全部成功，要么在事务回滚时所有的变更都挥毫撤销。

数据库事务，如InnoDB实现了的，拥有的属性统称为\*\*\*ACID\*\*\*，即原子性、一致性、隔离性与持久性。

参见 [ACID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_acid), [commit](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_commit), [isolation level](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_isolation_level), [lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_lock), [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback).

**transaction ID: 事务ID**

与每行关联的一个内部字段。这个字段可以在物理上可以被INSERT、UPDATE和DELETE操作变更以记录哪个事务已经锁住了这行。

参见 [implicit row lock](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_implicit_row_lock).

**transportable tablespace: 可传输表空间**

允许一个表空间（***tablespace***）从一个实例移动到另一个实例的特性。通常情况下，对于InnoDB表空间来说这是不可能的，因为所有的表数据是系统表空间（***system tablespace***）的一部分。在MySQL 5.6及更高版本中，FLUSH TABLE ... FROM EXPORT语法为一个InnoDB表向另一个服务拷贝做好准备；在其它服务器上执行ALTER TABLE ... DISCARD TABLESPACE和ALTER TABLE ... IMPORT TABLESPACE会让拷贝了的数据文件进入到另一个实例中。一个单独的.cfg文件与.ibd文件（\*\*\*.ibd file\*\*\*）一起拷贝，用来在导入时更新表的元数据（例如空间ID，***space ID***）。使用方法信息参见[第14.2.5.5节，拷贝表空间到另一个服务器（可传输表空间）][14.02.05.05]。

参见 [.ibd file](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ibd_file), [space ID](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_space_id), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_tablespace).

**.TRG file: 触发器参数文件**

**.TRN file 触发器命名空间信息文件**

**troubleshooting: 故障排除**

InnoDB可靠性与性能故障排除的资源包括：Information Schema表。

**truncate**

删除一个表中所有内容的DDL操作，同时表和相关的索引不受影响。相之相对的是drop。虽然在概念上它与没有WHERE子句的DELETE语句有相同的结果，但在实现上有着不同的操作：InnoDB创建一个新的空表，删掉旧表，然后再重命名新表替换掉老表。**因为这是一个DDL操作，所以不能回滚（*roll back*）**。

如果将要被truncate的表中包含引用另一个表的外键，truncate操作会使用一个较慢的操作：每次删除一行，以便根据任何ON DELETE CASCADE子句的需要删除引用表中的相应行。（MySQL 5.5及更高版本不允许这种慢的truncate，如果涉及外键，它会返回一个错误来替代。在这种情况下，请使用DELETE语句替代）。

**tuple: 元组**

指代一个有序集合的技术术语。它是个抽像的概念，用在正式的数据库理论的讨论中。在数据库领域中，元组常常被用表中一行的列来表示。它们也可以用来表示查询的结果集，例如，只获取表一部分列的查询或关联表中的列。

**two-phase commit: 两阶段提交**

XA规范下，分布式事务（***transaction***）中的一个操作。（有时候也简称为2PC）。当**多个数据库**参与到事务中时，要么所有的数据库提交（***commit***）都发生变更，要么所有的数据库都回滚变更（***rollback***）。

**U**

**undo: 撤销，或不译**

整个于事务（***transaction***）生命周期中的维护的数据，记录所有变更，这样它们可以在回滚（***rollback***）操作的情况下可以被撤销。它们存储于undo日志（***undo log***）中，也叫回滚段（***rollback segment***），要么在系统表空间（***system tablespace***）中，要么在单独的undo表空间（***undo tablespace***）中。

参见 [rollback](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback), [rollback segment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback_segment), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log), [undo tablespace].

**undo buffer: undo缓冲，不译**

参见 [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**undo log: undo 日志，不译**

保存被活动事务（***transaction***）修改了的数据拷贝的存储区。如果另一个事务需要查看原始数据（作为一致性读（***consistent read***）操作的一部分），没有修改的数据从这个存储区获取。

默认情况下，这个区域在物理上是系统表空间（***system tablespace***）的一部分。在MySQL 5.6及更高版本中，你可以使用[innodb\_undo\_tablespaces]和[innodb\_undo\_directory]配置选项来把它拆分到一个或多个独立的表空间（***tablespace***）文件中，即undo表空间（***undo tablespace***），可以选项性地放到另一种存储设备上，比如\*\*\*SSD\*\*\*。

The undo log is split into separate portions, the insert undo buffer and the update undo buffer. Collectively, these parts are also known as the rollback segment, a familiar term for Oracle DBAs. undo日志拆分为独立的部分，即插入undo缓冲（***insert undo buffer***）和更新undo缓冲（***update undo buffer***）。这些部分也统称为回滚段（***rollback segment***）中，这对于Oracle DBA来说是一个比较熟悉的术语。

参见 [consistent read](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_consistent_read), [rollback segment](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_rollback_segment), [SSD](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_ssd), [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [transaction](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_transaction), [undo tablespace].

**undo tablespace undo: 表空间**

在通过innodb\_undo\_tablespaces和innodb\_undo\_directory配置选项将undo日志从系统表空间（***system tablespace***）中分拆出来的情况下，包含undo日志（***undo log***）的一组文件。仅适用于MySQL 5.6及更高版。

参见 [system tablespace](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_system_tablespace), [undo log](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_undo_log).

**unique constraint: 唯一约束**

一列不能包含任何重复值的一种约束（***constraint***）断言。在关系（***relational***）代数的术语中，它常用来指一对一关系。为了高效检查一列是否可以被插入（也就是说，值不能已经存在于列中），唯一约束由唯一索引（***unique index***）支持。

参见 [constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_constraint), [relational](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_relational), [unique index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_index).

**unique index: 唯一索引**

在一个或一组具有唯一约束（***unique constraint***）的列上的索引。因为这个索引已知不含任何重复的值，所以某些类型的检索和计数操作要比普通类型的索引更高效。在此类型索引上的绝大多数检索是简单地判断某个值是否存在。这个索引中值的数量与表中的行数相同，或至少与关联的列中非空的行数相同。

插入缓冲（***insert buffering***）优化不会应用到唯一索引中。作为一种变通方法，你可以在往InnoDB表中加载数据时临时设置unique\_check=0。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality), [insert buffering](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_insert_buffering), [unique constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_constraint), [unique key](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_key).

**unique key: 唯一键**

包括一个唯一索引的列（一列或多列）集。当你可以定义正好匹配一行的WHERE条件，且查询可以用到相应的唯一索引（***unique index***）时，这个检索和错误处理起来就会非常高效。

参见 [cardinality](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_cardinality), [unique constraint](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_constraint), [unique index](https://github.com/xufubo/manual56/blob/master/docs/glossary.md#glos_unique_index).

**以下全看了**

**victim: 牺牲（死锁检测，牺牲影响最少行的事物）**

当检测到死锁时，自动选择回滚的事务。 InnoDB回滚已更新最少行的事务。可以使用innodb\_deadlock\_detect配置选项禁用死锁检测。

**wait: 等待**

当一个操作，比如获取一个锁（***lock***）、互斥锁（***mutex***）或闩锁（***latch***）时，不能立即完成，InnoDB暂停并再次尝试。

在高负载和多事务的系统上，你可以使用SHOW INNODB STATUS命令或者***Performance Schema***的输出来判断线程的大量时间是否是由于等待而被挂起，以及如何提高并发（***concurrency***）能力。更多信息：

[concurrency](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_concurrency), [latch](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_latch), [lock](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_lock), [mutex](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_mutex), [Performance Schema](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_performance_schema), [spin](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_spin)

**warm backup: 温备**

在数据库运行时发生的备份（***backup***），但备份期间它限制了一些数据库的操作。例如，表可能变成只读。对于繁忙的应用和网站，你可能优先使用热备（***hot backup***）。

**warm up: 预热**

服务器启动后，在一个普通的负载（***workload***）下让系统运行一段时间，以便buffer pool和其它内存区域像在平时一样被填充。 这个过程在MySQL服务器重启或在一个新的负载下自然产生。注意，通产在进行性能测试之前应该进行预热。自MySQL 5.6开始，你可以通过设置配置变量innodb\_buffer\_pool\_dump\_at\_shutdown=ON和innodb\_buffer\_pool\_load\_at\_startup=ON来加速预热。在MySQL5.7中这些选项默认是启用的。

**workload: 工作负载**

**write combining: 合并写**

当脏页（***dirty page***）从InnoDB buffer pool 刷新（***flush***）时为了减少写操作的优化技术。如果页中的一行被更新了多次，或同一个页中的多行被更新，它们的所有的更新会在一次写操作中存储到数据文件中，而不是每个变更写一次。这就大大减少了IO次数。

**XA: XA**

一个用来协调分布式事务（***transaction***）的标准接口，允许**多个数据库**在保持ACID的同时参与到一个事务中来。分布式事务的支持默认是打开的。如果你用不上这个特性，你可以使用innodb\_support\_xa配置选项来禁用它，避免对每个事务额外的fsync带来的性能负载。从MySQL 5.7.10起，禁用innodb\_support\_xa是不允许的，因为它使复制不安全，并防止与二进制日志组提交相关联的性能提升。 innodb\_support\_xa配置选项在MySQL 8.0中被删除。

**young InnoDB Buffer Pool**

InnoDB buffer pool中页面（***page***）的特性，意味着它刚刚被访问过，这样它就不会很快被LRU算法刷新（***flush***）。这个术语也用在information schema中的与buffer pool相关的表的列名上。