一、Mysql两种常用存储引擎myisam和innodb的文件存储结构

**1. 区别：**

myisam 不支持事物，只支持表级锁，不支持外键，物理结构（三个文件），非聚簇索引。

innodb 支持事务和行级锁，支持外键，物理结构（两个文件），聚簇索引。

如果系统读多写少，对原子性要求低，那么MyISAM最好的选择。且MyISAM恢复速度快，可直接用备份覆盖恢复。如果系统读少写多的时候，尤其是并发写入高的时候，InnoDB就是首选了。

**2. myism物理文件结构为：**

.frm文件：与表相关的元数据信息都存放在frm文件，包括表结构的定义信息等。

.myd文件：用于存储myisam表的数据

.myi文件：用于存储myisam表的索引相关信息

**3. innodb的物理文件结构为：**

.frm文件：与表相关的元数据信息都存放在frm文件，包括表结构的定义信息等。

.ibd文件和.ibdata文件：

这两种文件都是存放innodb数据的文件，之所以用两种文件来存放innodb的数据，是因为innodb的数据存储方式能够通过配置来决定是使用**共享表空间**存放存储数据，还是用**独享表空间**存放存储数据。

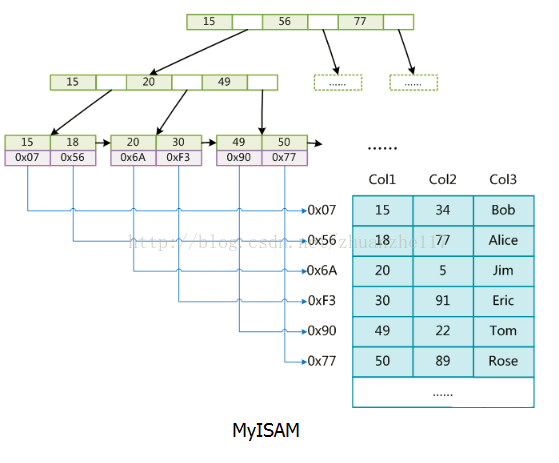
独享表空间存储方式使用.ibd文件，并且每个表一个ibd文件

共享表空间存储方式使用.ibdata文件，所有表共同使用一个ibdata文件

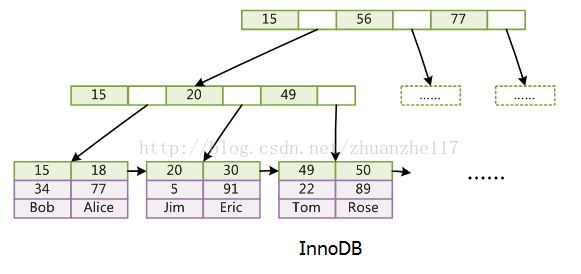
使用哪种方式的参数在mysql的配置文件中 innodb\_file\_per\_table

**4.索引不同**

（1）myisam（data存的是数据地址。索引是索引，数据是数据。）



（2）innodb（data存的是数据本身，索引也是数据。）



索引的总结：

从历史上来说MyISAM历史更加久远，所以InnoDB性能也就更好了，在这我们需要考虑当我们修改数据库中的表的时候，数据库发生了变化，那么他们的主键的地址也就发生了变化，这样你的MyISAM的主索引和辅助索引就需要重新建立索引。而InnoDB只需要改变主索引，因为它的辅助索引是存主键的，所以这样考虑InnoDB更加高效。

**5. 常见问题**

**5.1 数据库如何应对大规模写入和读取**

（1）NoSQL它通过降低数据的安全性，减少对事务的支持，减少对复杂查询的支持，

来获取性能上的提升。

（2）缓存和读写分离

（3）分库分表

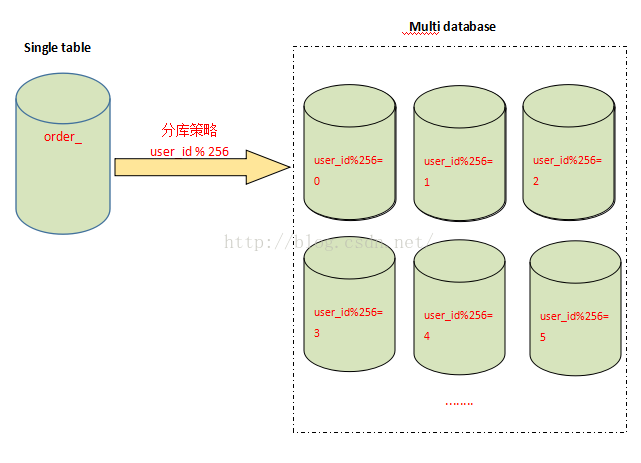
水平切分：不修改数据库表结构，通过对表中数据的拆分而达到分片的目的，一般水平拆分在查询数据库的时候可能会用到 union 操作（多个结果并）。

可以根据hash或者日期进行进行分表。

垂直切分：修改表结构，按照访问的差异将某些列拆分出去，一般查询数据的时候可能会用到 join 操作；把常用的字段放一个表,不常用的放一个表，把字段比较大的比如text的字段拆出来放一个表里面。

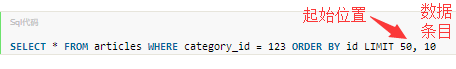
分库：分表能够解决单表数据量过大带来的查询效率下降的问题，但是，却无法给数据库的**并发处理**能力带来质的提升。

面对高并发的读写访问，当数据库 master 服务器无法承载写操作压力时，不管如何扩展 slave 服务器，此时都没有意义了。这个时候就需要对数据库进行拆分，从而提高数据库写入能力，这就是所谓的分库。分库可以采用对一个**关键字取模**的方式，来对数据访问进行路由。



**大量数据如何分页显示：**

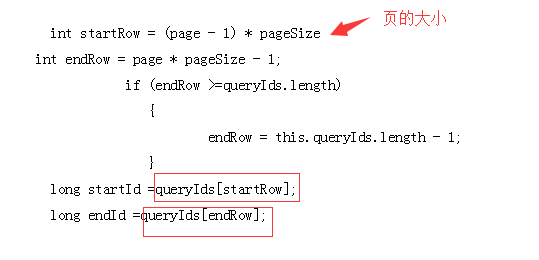
（1）在千万级分页时使用 limit



（2）设置一个自增的字段

在待查询的数据库表上增加一个用于查询的自增字段（比如“queryId”），先按照大小顺序的倒序查出所有的queryId，因为只是查询queryId字段，即使表格中的数据量很大，该查询也会很快得到结果。然后将得到的queryId保存在应用服务器的一个数组中。

用户在客户端进行翻页操作时，客户端将待查询的 页号 page 作为参数传递给应用服务器，服务器通过页号和queyId数组算出待查询的queyId最大和最小值，然后进行查询。



**5.2 读写分离（MySql Proxy可以实现）**

读写分离就是在主服务器上修改数据，数据会同步到从服务器，从服务器只能提供读取数据，不能写入，实现备份的同时也实现了数据库性能的优化，以及提升了服务器安全。

**如何保证数据一致性：**

（1）主节点

保证每次事务提交后，都能实时刷新到磁盘中，尤其是确保每次事务对应的binlog都能及时刷新到磁盘中，只要有了binlog，InnoDB就有办法做数据恢复，不至于导致主从复制的数据丢失。

（2）从节点

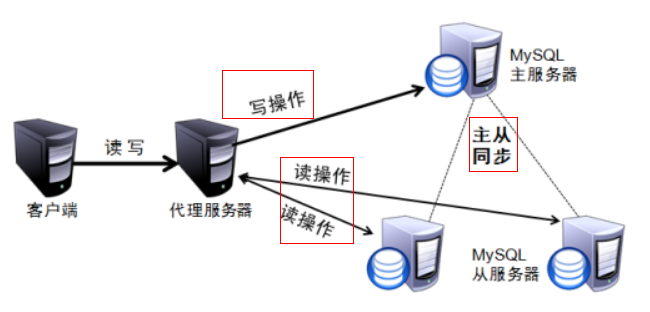
开启 relay log 自动修复机制，发生 crash 时，会自动判断哪些 relay log 需要重新从master 上抓取回来再次应用，以此避免部分数据丢失的可能性。

**优点：**

1.增加冗余

2.增加了机器的处理能力

3.对于读操作为主的应用，使用读写分离是最好的场景，因为可以确保写的服务器压力更小，而读又可以接受点时间上的延迟。



**5.3 数据库的缓存重建**

（1）用一些可以提供持久化功能的缓存来实现，比如Redis

（2）MongoDB与上面的方式不太一样，MongoDB采用mmap来将数据文件映射到内存中，所以当MongoDB重启时，这些映射的内存并不会清掉，因为它们是由操作系统维护的（所以当操作系统重启时，MongoDB才会有相同问题）。相对于其它一些自己维护Cache的数据库，MongoDB在重启后并不需要进行缓存重建与预热。

（3）缓存重建加锁的方式，也能部分解决此问题。简单来说就是缓存重建时，当多个客户端对同一个缓存数据发起请求时，会在客户端采用加锁等待的方式，对同一个Cache的重建需要获取到相应的锁才行，只有一个客户端能拿到锁，并且只有拿到锁的客户端才能访问数据库重建缓存，其它的客户端都需要等待这个拿到锁的客户端重建好缓存后直接读缓存，其结果是对同一个缓存数据，只进行一次数据库重建访问。但是如果访问分散比较严重，还是会瞬间对数据库造成非常大的压力。

**5.4 mysql5.6和5.7的区别**

安全性

* 用户表 mysql.user 的 plugin字段不允许为空， 默认值是 mysql\_native\_password，而不是 mysql\_old\_password，不再支持旧密码格式；
* 增加密码过期机制，过期后需要修改密码，否则可能会被禁用，或者进入沙箱模式；
* 增加密码过期机制，过期后需要修改密码，否则可能会被禁用，或者进入沙箱模式；
* 提供了更为简单SSL安全访问配置，并且默认连接就采用SSL的加密方式。

灵活性

* MySQL数据库从5.7.8版本开始，也提供了对JSON的支持。
* 可以混合存储结构化数据和非结构化数据，同时拥有关系型数据库和非关系型数据库的优点
* 能够提供完整的事务支持
* generated column是MySQL 5.7引入的新特性，所谓generated column，就是数据库中这一列由其他列计算而得

易用性

* 在MySQL 5.7 之前，如果用户输入了错误的SQL语句，按下 ctrl+c ，虽然能够”结束”SQL语句的运行，但是，也会退出当前会话，MySQL 5.7对这一违反直觉的地方进行了改进，不再退出会话。
* MySQL 5.7可以explain一个正在运行的SQL，这对于DBA分析运行时间较长的语句将会非常有用。
* sys schema是MySQL 5.7.7中引入的一个系统库，包含了一系列视图、函数和存储过程， 该项目专注于MySQL的易用性。

例如：如何查看数据库中的冗余索引；如何获取未使用的索引；如何查看使用全表扫描的SQL语句。

可用性

* 在线设置 复制的过滤规则 不再需要重启MySQL，只需要停止SQLthread，修改完成以后，启动SQLthread。
* 在线修改buffer pool的大小。
* Online DDL MySQL 5.7支持重命名索引和修改varchar的大小，这两项操作在之前的版本中，都需要重建索引或表。
* 在线开启GTID ，在之前的版本中，由于不支持在线开启GTID，用户如果希望将低版本的数据库升级到支持GTID的数据库版本，需要先关闭数据库，再以GTID模式启动，所以导致升级起来特别麻烦。

性能

* 临时表的性能改进。

临时表只在当前会话中可见

临时表的生命周期是当前连接（MySQL宕机或重启，则当前连接结束）

* 只读事务性能改进。

MySQL 5.7通过 避免为只读事务分配事务ID ，不为只读事务分配回滚段，减少锁竞争等多种方式，优化了只读事务的开销，提高了数据库的整体性能。

* 加速连接处理。

在MySQL 5.7之前，变量的初始化操作（THD、VIO）都是在连接接收线程里面完成的，现在将这些工作下发给工作线程，以减少连接接收线程的工作量，提高连接的处理速度。这个优化对那些频繁建立短连接的应用，将会非常有用。

* 复制性能的改进 （支持多线程复制（Multi-Threaded Slaves, 简称MTS）

MySQL的默认配置是库级别的并行复制，为了充分发挥MySQL 5.7的并行复制的功能，我们需要将slave-parallel-type配置成LOGICAL\_CLOCK。

* 支持多源复制（Multi-source replication）

严格性改变

* 默认启用 STRICT\_TRANS\_TABLES 模式。
* 对 ONLY\_FULL\_GROUP\_BY 模式实现了更复杂的特性支持，并且也被默认启用。
* 其他被默认启用的sql mode还有 NO\_ENGINE\_SUBSTITUTION。

默认参数的改变

* 默认binlog格式调整为ROW格式
* 默认binlog错误后的操作调整为ABORT\_SERVER

在先前的选项下（binlog\_error\_action=IGNORE\_ERROR），如果一个错误发生，导致无法写入binlog，mysql-server会在错误日志中记录错误并强制关闭binlog功能。这会使mysql-server在不记录binlog的模式下继续运行，导致从库无法继续获取到主库的binlog。