存储引擎 是否支持事物 锁级别 是否支持在线非阻塞备份

MyISAM N TABLE N

InnoDB Y ROW Y

Maria Y ROW N

MySQL的存储引擎：

MyISAM:

非事务

不支持外键

支持全文索引

没有数据缓存，只缓存索引(只缓存打开表的文件描述符，但是在服务器级别是支持缓存的，缺陷不大)

支持RTREE索引； 不支持HASH索引

默认BTREE索引

支持表级别的锁

读操作效率高，适用于读多写少的场景

支持数据压缩(主要目的用于数据仓库)，myisampack

数据存储每表三个文件(.frm, .MYD, .MYI)

MySQL能对MyISAM引擎表进行自动检查和自动修复，也能手工修复(CHECK TABLE tb\_name和REPAIR TABLE tb\_name,外部工具myisamchk)

延迟更新索引(delay\_key\_write)

使用mysqlhotcopy在线备份(不是非阻塞的)；每个表支持最多64个索引

key\_buffer\_size:

InnoDB: -->XtraDB(Percona, Xtraback)

运行MVCC、事务

行级锁

缓存数据和索引，大小可配置

支持外键

支持**聚簇**BTREE索引(每表只能一个)支持自适应的HASH

支持热备

基于表空间存储数据(也支持每表一个表空间)

自适应的hash索引(非显式)

更快的基于主键的读取(纯粹的读取不比MyISAM好)。

支持可配置的索引缓存和数据缓存大小

innodb\_data\_home\_dir：表空间文件的存储目录

innodb\_data\_file\_path：表空间文件的路径，初始大小，能否自动增长，一次增长多大

innodb\_log\_group\_home\_dir： innodb事务日志文件的路径

事务日志能够将随机IO转为顺序IO

innodb\_file\_per\_table

innodb\_log\_files\_in\_group: 日志文件的个数，默认为2个；

innodb\_log\_file\_size：日志文件的大小；默认为5M；

innodb\_log\_buffer\_size：日志缓冲空间

innodb\_flush\_log\_at\_trx\_commit：事务提交是不是立即同步事务日志中的数据至表空间文件中；

innodb\_flush\_method

innodb\_buffer\_pool\_size: 缓存空间大小

Memory (heap:堆)

临时表

HASH索引

Archive：

只支持INSERT和SELECT，不支持索引

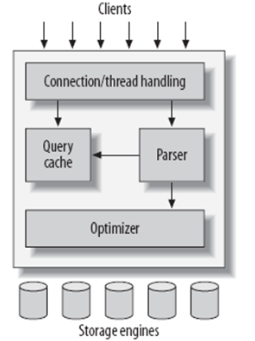
CSV:(逗号分隔值)

数据存储为纯文本文件

Blackhole：黑洞

NDB Cluster

SolidDB,PBXT,Maria,Falcon



有些配置是在存储引擎上配置的，有的是在存储引擎级别配置的。所以，有些特性上面提供，有些下面提供，有时候都提供。

并发控制：

MVCC：

多版本并发控制

降低MySQL操作时加锁需要

对每一行都施加了系统版本号：读，写

每一个事务开始时，系统版本号自动加1；

10，START TRANSACTION

11

锁分类：

读锁：共享锁，读时不能被修改，但可以同时读；写饥饿

写锁：独占锁；排他锁

锁粒度：

表级锁：

页：存储引擎管理数据块按照页来管理；

行

锁发起的方式：

隐式锁 需要时自动加锁

显式锁：

服务器级别的锁可以直接指定：

FLUSH TABLES WITH READ LOCK; //读锁

LOCK TABLE course WRITE; //写锁

UNLOCK TABLES;

一组同时执行或者不执行的SQL就是**事物，满足四个条件：ACID**

原子性：一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

一致性：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的默认规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

隔离性：当两个或者多个事务并发访问（此处访问指查询和修改的操作）数据库的同一数据时所表现出的相互关系。**事务隔离分为不同级别**，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。

持久性：在事务完成以后，该事务对数据库所作的更改便持久地保存在数据库之中，并且是完全的。

**隔离级别：**一次事物中修改的未提交事物能否被别的用户操作看到，完全隔离性能极低

READ UNCOMMITTED (读未提交)： 脏读，未提交的可以被看见

READ COMMITTED (读提交)：不可重复读，不提交的看不见

REPEATABLE READ (可重读): 幻读。**默认级别**；各自的相互独立，自己原来是什么样就是什么样

SERIALIZABLE (串行化) 最高级别；必须别人提交之后才能读。效率极低

<http://xm-king.iteye.com/blog/770721> 解释的很清楚

查看现在的隔离级别： SHOW VARIABLES LIKE ‘tx\_’

设定隔离级别：全局的：SET GLOBAL TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

退出生效；这样就设置为读未提交级别了。

会话级别的修改：

隔离级别就是某用户修改，另一个用户是否也能看见修改。别人未提交的能被看见就是读未提交。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

开始一个事物：START TRANSACTION

提交 COMMIT

回滚 ROLLBACK

保存点 SAVE POINT SP\_NAME

回滚到某个点 ROLLBACK SP\_NAME

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

至于怎么做到的默认隔离级别下不同的读到不同的配置这就是MVCC多版本并发控制；

一个事物启动，为他创建一个快照，

<http://baike.baidu.com/view/1887040.htm>