**MyISAM**

1. 不支持事务、外键和行级锁,适合小项目

优点:1.读多写少的项目，效率高2.没有磁盘碎片，节省空间

**存储引擎**

影响引擎性能的因素：数据大小，I/O请求量，主键还是二级索引

**InnoDB：**优先选择InnoDB，除非用到InnoDB不具备的特性，没有其他办法替代。

1. 事务2.热备份3.崩溃恢复4.聚簇索引

**聚簇索引**:按物理位置顺序存放，引擎缓存索引和数据

插入慢，排序快。非聚簇索引（二级索引）则没有按序存放，需要额外消耗资源来排序。

**MyISAM：**一般读写性能高，但随着应用压力上升，各种锁争用，崩溃数据丢失问题

1. MyISAM引擎缓存索引，而不缓存数据

**MySQL性能优化**

索引优化 查询优化 库表结构优化

**索引优化**

**索引**

1）索引大大减少了服务器需要扫描的数据量

2）索引可以帮助服务器避免排序和临时表

3）索引可以将随机I/O变成顺序I/O

一般oracle会判断,使用索引返回的数据量不超过总表的5%时,才走索引..否则就是全表扫描

**查询优化**

**1.优化数据访问**

查询性能低下最基本原因是访问数据太多。

**1)应用服务**

* 确认应用服务是否检索大量查过需要的数据（访问太多的行或列）

加limit

* 返回所有列或者多表关联返回需要的列
* 重复查询相同的数据：缓存

**2)MySQL服务器**

确认MySQL服务器是否在分析大量超过需要的数据行

**响应时间**

**扫描行数和返回行数**

**扫描行数和访问类型**

ALL-index-range-ref

一般MYSQL能够使用如下三种方式应用where条件记录，从好到坏依次为

1) 在存储索引层，在索引中使用where条件来过滤不匹配的记录。

2) 在mysql服务器层, 扫描覆盖索引过滤并返回记录。无需在回表查询记录(Using index)。

3) 在mysql服务器层，先从数据表读出记录然后过滤(Using Where)

如果发现查询需要扫描大量的数据但返回少数的行，那么通常可以尝试下面的技巧：

1)使用索引覆盖扫描，把所有需要的列都放到索引中，这样存储引擎无须返回表获取对应行就可以返回结果了。

2)改变库表结构，使用单独的汇总表。

3)重写这个复杂的查询

**2.重构查询方式**

**1)复杂查询还是简单查询**

MySQL吞吐量：10w/s查询

扫描内存：100+w行/s

**2)分解关联查询**

**单表 vs 联表**

看具体业务场景。没到阀值前，都可以，有的业务也许永远到达这个阀值。

高并发高读环境：单表or冗余，可做缓存。建覆盖索引。

联表后期扩展和维护工作量大，简单说dao层的复用率低。

**3.优化特定类型的查询**

**Count（）**

1.MyISAM 2. 索引覆盖扫描3.汇总表 4.增加memcached

**关联查询**

1. ON或using列有索引,一般在连表顺序第二个create index。
2. Group by和Order By之设计一个表的列

**子查询：**尽量 关联子查询替代，当前MySQL版本需要

GROUP BY和DISTINCT

LIMIT：一般ORDER BY …LIMIT

**库表结构优化**

1. **选择合适的数据类型**

**更小的通常更好**：占用更少的磁盘、内存和CPU缓存，CUP处理周期

**简单就好**：简单类型CPU处理快。例如整形比字符操作代价低

**尽量避免NULL**：NULL使索引更复杂，需要更多存储空间

1. **范式和反范式**

范式需要联表，代价昂贵。还可能使一些索引策略无效，比如覆盖索引。

1. **缓存表和汇总表**

冗余是在一张表。

汇总表：group by 或count

缓存表：存储从schema中获取比较慢的数据表。

例如主键+部分列。用缓存或solr(搜索效率更高)

**优点**：避免复杂，昂贵的实时跟新操作

[**MYSQL数据库优化**](http://blog.csdn.net/dongnan591172113/article/details/52174619)

不同业务场景优化的策略不同，比如数据量、访问量（QPS每秒查询率）、并发量、读写比例、数据是否快速增长、数据需不需要强一致

优化SQL和索引-> 缓存-> 主从复制或主主复制-> 分区表-> 垂直拆分-> 水平切分-> 选

**避免扫描大量的数据但只返回少量的行**

1.覆盖索引

2.改变表结构，例如汇总表

3.重写复杂的查询

4.定期备份，删除

**减少索引和数据的碎片**

Optimize table;