**MYSQL**

Query Cache: QCache\_hit远大于QCache\_insert才有意义，适合应用更新少。一般在应用中做缓存。

**性能指标**：

1. 吞吐量2.响应时间3.并发性

**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

[**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-> 选择不同的存储引擎

1. 反范式(冗余)：避免连表
2. 缓存表和汇总表（1.goup by 2.count(\*)）

**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

多列索引

**SQL优化**

1.定位执行效率低的SQL 1)慢查询2)show processlist查看线程状态

2.Explain 分析慢SQL的执行计划,并采取对应措施

**减少索引和数据的碎片**

Optimize table;

**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

**查询性能优化**

分解关联查询：1.连表

**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

**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