## 如何写出高效的SQL语句（从建表到查询一步一步往后说）

### 建表时可以得出以下优化建议

1. 数据类型尽量用数字型，数字型比字符型的快
2. 选择正确的表引擎

MyISAM 适合于一些需要大量查询的应用，但其对于有大量写操作并不是很好。甚至你只是需要update一个字段，整个表都会被锁起来，而别的进程，就算是读进程都无法操作直到读操作完成。另外，MyISAM 对于 SELECT COUNT(\*) 这类的计算是超快无比的。   
 InnoDB 的趋势会是一个非常复杂的存储引擎，对于一些小的应用，它会比 MyISAM 还慢。他是它支持“行锁” ，于是在写操作比较多的时候，会更优秀。并且，他还支持更多的高级应用，比如：事务。

1. 选择合适的数据类型。

如果你有一个字段，比如“性别”，“国家”，“民族”，“状态”或“部门”，你知道这些字段的取值是有限而且固定的，那么，你应该使用 ENUM 而不是 VARCHAR。

1. 尽量给字段加上NOT NULL
2. 一个表不要加太多索引，因为索引影响插入和更新的速度
3. 适当的使用冗余的反范式设计，以空间换时间有的时候会很高效

### 查询的时候得出的优化建议：

1. 尽量不要在数据库中做运算
2. 使用预处理语句。例如使用PDO来操作mysql

在性能方面，当一个相同的查询被使用多次的时候，这会为你带来可观的性能优势。你可以给这些Prepared Statements(预处理语句)定义一些参数，而MySQL只会解析一次。

1. 不要在生产环境程序中使用select \* from 的形式查询数据。只查询需要使用的列
2. 查询尽可能使用limit减少返回的行数，减少数据传输时间和带宽浪费
3. 所有的SQL关键字用大写，避免SQL语句重复编译造成系统资源的浪费

12、开启慢查询日志，定期用explain或desc优化慢查询中的SQL语句

13、最先出现的条件，一定是过滤和排除掉更多结果的条件，第二出现的次

之。

14、对查询进行优化，尽量避免全表扫描。首先应考虑在where以及order by涉及的列上建立索引。

15、尽量避免在where子句中对字段进行null值判断。这会进行全表扫描

Select id,name from user where name is null;

1. 尽量避免在where子句中对字段进行表达式操作。这会导致引擎放弃使用索引而进行全表扫描

SELECT id,name FROM user where age/12;

1. 使用连接（join）代替子查询select \* from customInfo where customId in ( select customId from saleInfo )

18、对于OR子句，如果要利用索引，则OR之间的每个条件列都必须用到索引，如果没有索引，则应该考虑增加索引。