索引优化：

1. 独立的列：在进行查询时，索引列不能是表达式的一部分，也不能是函数的参数，否则无法使用索引。
2. 多列索引：在需要使用多个列作为条件进行查询时，使用多列索引比使用多个单列索引性能好。
3. 让选择性最强的索引列放在前面
4. 索引包含所有需要查询的字段的值。

索引的优点

大大减少了服务器需要扫描的数据行数。

帮助服务器避免进行排序和分组，以及避免创建临时表（B+Tree 索引是有序的，可以用于 ORDER BY 和

GROUP BY 操作。临时表主要是在排序和分组过程中创建，不需要排序和分组，也就不需要创建临时表）。

将随机 I/O 变为顺序 I/O（B+Tree 索引是有序的，会将相邻的数据都存储在一起）。

索引的使用条件

对于非常小的表、大部分情况下简单的全表扫描比建立索引更高效；

对于中到大型的表，索引就非常有效；

但是对于特大型的表，建立和维护索引的代价将会随之增长。这种情况下，需要用到一种技术可以直接区分出

1. 需要查询的一组数据，而不是一条记录一条记录地匹配，例如可以使用分区技术。

查询性能优化

Explain 用来分析 SELECT 查询语句，开发人员可以通过分析 Explain 结果来优化查询语句。

比较重要的字段有：

select\_type : 查询类型，有简单查询、联合查询、子查询等

key : 使用的索引；rows : 扫描的行数

1. 减少请求的数据量

只返回必要的列：最好不要使用 SELECT \* 语句。

只返回必要的行：使用 LIMIT 语句来限制返回的数据。

缓存重复查询的数据：使用缓存可以避免在数据库中进行查询，特别在要查询的数据经常被重复查询时，缓存带来的查询性能提升将会是非常明显的。

2. 减少服务器端扫描的行数

最有效的方式是使用索引来覆盖查询。

3 切分大查询：一个大查询如果一次性执行的话，可能一次锁住很多数据、占满整个事务日志、耗尽系统资源、阻塞很多小的但重要的查询。

4 分解大连接查询

将一个大连接查询分解成对每一个表进行一次单表查询，然后在应用程序中进行关联