**字段类型的优化（适用于记录条数较多，数据量较大的场景）**

**1.数字类型：**

a.非万不得已不要使用DOUBLE，不仅仅只是存储长度的问题，同时还会存在精确性的问题。

b.固定精度的小数，也不建议使用DECIMAL，建议乘以固定倍数转换成整数存储，可以大大节省存储空间，且不会带来任何附加维护成本。

c.对于整数的存储，数据量较大的建议区分开 TINYINT / INT / BIGINT 的选择，因为三者所占用的存储空间也有很大的差别，能确定不会使用负数的字段，建议添加unsigned定义。

**2.字符类型：**

a.非万不得已不要使用 TEXT 数据类型，其处理方式决定了他的性能要低于char或者是varchar类型的处理。

b.定长字段，建议使用 CHAR 类型

c.不定长字段尽量使用 VARCHAR，且仅仅设定适当的最大长度。

**3.时间类型：**

a.尽量使用TIMESTAMP类型，因为其存储空间只需要 DATETIME 类型的一半。

b.对于只需要精确到某一天的数据类型，建议使用DATE类型，因为他的存储空间只需要3个字节，比TIMESTAMP还少。

c.不建议通过INT类型存储一个unix timestamp 的值，因为这太不直观，会给维护带来不必要的麻烦，同时还不会带来任何好处。

**4.ENUM & SET：（都属于字符串类型）**

a.对于状态字段，可以尝试使用 ENUM 来存放，因为可以极大的降低存储空间，而且即使需要增加新的类型，只要增加于末尾，修改结构也不需要重建表数据。

b.如果是存放可预先定义的属性数据，可以尝试使用SET类型，即使存在多种属性，同样可以游刃有余，同时还可以节省不小的存储空间

**5.LOB类型：**

强烈反对在数据库中存放 LOB 类型数据

**字符编码**

**字符集直接决定了数据在MySQL中的存储编码方式，由于同样的内容使用不同字符集表示所占用的空间大小会有较大的差异，所以通过使用合适的字符集，可以帮助我们尽可能减少数据量，进而减少IO操作次数。**

1.纯拉丁字符能表示的内容，没必要选择 latin1 之外的其他字符编码，因为会节省大量的存储空间

2.如果我们可以确定不需要存放多种语言，就没必要非得使用UTF8或者其他UNICODE字符类型，这回造成大量的存储空间浪费。

3.MySQL的数据类型可以精确到字段，所以当我们需要大型数据库中存放多字节数据的时候，可以通过对不同表不同字段使用不同的数据类型来较大程度减小数据存储量，进而降低 IO 操作次数并提高缓存命中率。

**适当拆分**

**有些时候，我们可能会希望将一个完整的对象对应于一张数据库表，这对于应用程序开发来说是很有好的，但是有些时候可能会在性能上带来较大的问题。**

当我们的表中存在类似于 TEXT 或者是很大的 VARCHAR类型的大字段的时候，如果我们大部分访问这张表的时候都不需要这个字段，我们就该义无反顾的将其拆分到另外的独立表中，以减少常用数据所占用的存储空间。这样做的一个明显好处就是每个数据块中可以存储的数据条数可以大大增加，既减少物理 IO 次数，也能大大提高内存中的缓存命中率。

上面几点的优化都是为了减少每条记录的存储空间大小，让每个数据库中能够存储更多的记录条数，以达到减少 IO 操作次数，提高缓存命中率。下面这个优化建议可能很多开发人员都会觉得不太理解，因为这是典型的反范式设计，而且也和上面的几点优化建议的目标相违背。

**适度冗余**

**这样做是会增大每条记录的大小，降低每条记录中可存放数据的条数，但是在有些场景下我们仍然还是不得不这样做：**

1.被频繁引用且只能通过 Join 2张(或者更多)大表的方式才能得到的独立小字段。

2.这样的场景由于每次Join仅仅只是为了取得某个小字段的值，Join到的记录又大，会造成大量不必要的 IO，完全可以通过空间换取时间的方式来优化。不过，冗余的同时需要确保数据的一致性不会遭到破坏，确保更新的同时冗余字段也被更新。

**尽量使用 NOT NULL(就是尽量少使用NULL)**

NULL 类型比较特殊，SQL 难优化。虽然 MySQL NULL类型和 Oracle 的NULL 有差异，会进入索引中，但如果是一个组合索引，那么这个NULL 类型的字段会极大影响整个索引的效率。此外，NULL 在索引中的处理也是特殊的，也会占用额外的存放空间。

很多人觉得 NULL 会节省一些空间，所以尽量让NULL来达到节省IO的目的，但是大部分时候这会适得其反，虽然空间上可能确实有一定节省，倒是带来了很多其他的优化问题，不但没有将IO量省下来，反而加大了SQL的IO量。所以尽量确保 DEFAULT 值不是 NULL，也是一个很好的表结构设计优化习惯

**读写分离（主库只写和少量实时读取请求）**

**快速更新大数据库表禁止直接运行count(\*)统计**

**尽量避免在列上运算，这样会导致索引失效。**

**日期运算**

优化前：

select \* from system\_user where date(createtime) >= '2015-06-01'

优化后：

select \* from system\_user where createtime >= '2015-06-01'

**加，减，乘，除**

优化前：

select \* from system\_user where age + 10 >= 20

优化后：

select \* from system\_user where age >= 10

**用整型设计索引**

**占用的字节少，相对与字符串索引要快的多。特别是创建主键索引和唯一索引的时候。**

1）设计日期时候，建议用int取代char(8)。例如整型：20150603。 2）设计IP时候可以用bigint把IP转化为长整型存储。



**join时，使用小结果集驱动大结果集**

使用join的时候，应该尽量让小结果集驱动大的结果集，把复杂的join查询拆分成多个query。因为join多个表的时候，可能会有表的锁定和阻塞。如果大结果集非常大，而且被锁了，那么这个语句会一直等待。这个也是新手常犯的一个错误！

优化前：

select \* from table\_a a left join table\_b b on a.id = b.id left join table\_c c

on a.id = c.id where a.id > 100 and b.id < 200

优化后：

select \* from ( select \* from table\_a where id > 100) a
left join( select \* from table\_b where id < 200) b
on a.id = b.id left join table\_c on a.id = c.id

**仅列出需要查询的字段**

仅列出需要查询的字段，新手一般都查询的时候都是\*，其实这样不好。这对速度不会有明显的影响，主要考虑的是节省内存。

优化前：

select \* from system\_user where age > 10

优化后：

select username,email from system\_user where age > 10

**使用批量插入节省交互**

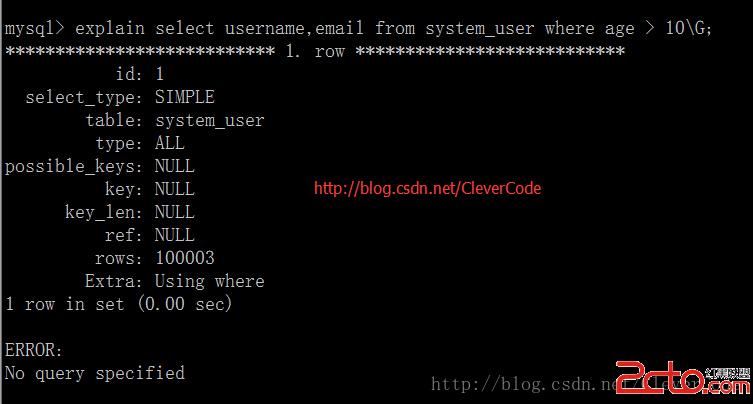
优化前：

insert into system\_user(username,passwd) values('test1','123456')
insert into system\_user(username,passwd) values('test2','123456')
insert into system\_user(username,passwd) values('test3','123456')

优化后：

insert into system\_user(username,passwd) values('test1','123456'),('test2','123456')，('test3','123456')

**多习惯使用explain分析sql语句**



**使用profiling分析sql语句时间开销**

1.将经常用到的字段（比如经常要用这些字段来排序、用来做搜索），最好将这些字段设为索引。

2.字段的种类尽可能用int 或者tinyint类型。另外字段尽可能用NOT NULL。

3.当然无可避免某些字段会用到text ,varchar等字符类型，最好将text字段单独另外一个表存储（用主键关联好）

4.字段的类型，以及长度，是一个很考究开发者优化功力的一个方面。如果表数据有一定的量了，不妨用PROCEDURE ANALYSE()命令来取得字段的优化建议！（在phpmyadmin里可以在查看表时，点击 “Propose table structure” 来查看这些建议） 如此可以让你的表字段结构 趋向完善。

5.select \* 尽量少用，你想要什么字段 就select 什么字段出来 不要老是用\* 号！同理，只要一行数据时尽量使用 LIMIT 1

6.绝对不要轻易用order by rand() ，很可能会导致mysql的灾难！！

7.每个表都应该设置一个ID主键，最好的是一个INT型，并且设置上自动增加，AUTO\_INCREMENT标志，这点其实应该作为设计表结构的第一件必然要做的事！！

8.拆分大的 DELETE 或 INSERT 语句。因为这两个操作是会锁表的，表一锁住了，别的操作都进不来了，就我来说 有时候我宁愿用for循环来一个个执行这些操作。

9.不要用永久链接 mysql\_pconnect()；除非你真的非常肯定你的程序不会发生意外，不然很可能也会导致你的mysql死掉。

10.永远别要用复杂的mysql语句来显示你的聪明。就我来说，看到一次关联了三，四个表的语句，只会让人觉得很不靠谱。