1.慢查询

当一个资源变得效率低下的时候，应该了解一下为什么会这样。有如下可能原因：
1.资源被过度使用，余量已经不足以正常工作。
2.资源没有被正确配置
3.资源已经损坏或者失灵
因为慢查询，太多查询的时间过长而导致堆积在逻辑上。
慢查询到底是原因还是结果？在深入调查前是无法知晓的。记住，在正常的时候这个查询也是正常运行的。一个查询需要filesort和创建临时表并不一定意味着就是有问题的。尽管消除filesort和临时表通常来说是“最佳实践”。

2.MySQL数据类型

> 更小的通常更好：一般情况下，应该尽量使用可以正确存储数据的最小数据类型。更小的数据类型通常更快，因为它们占用更少的磁盘、内存和cpu缓存，并且处理时需要的cpu周期也更少。
>
> 简单就好：简单数据类型的操作通常需要更少的cpu周期，例如整型比字符操作代价更低，因为字符集和校对规则（排序规则）使字符比较比整型比较更复杂，使用整型存储IP地址。
>
> 尽量避免NULL：通常情况下最好制定列为NOT NULL，除非真的需要存储NULL值。如果查询中包含可为NULL的列，对mysql来说更难优化，因为可为NULL的列使得索引、索引统计和值比较都更复杂。
>
> 很多MySQL的数据类型可以存储相同类型的数据，只是存储的长度和范围不一样、允许的精度不同，或者需要的物理空间（磁盘和内存空间）不同。

别名

> MYSQL为了兼容性支持很多别名，例如integer、bool、numeric，他们都只是别名，虽然可能令人不解，但是不会影响性能。如果建表的时候采用数据类型的别名，然后用show create table检查，会发现mysql报告的是基本类型，而不是别名。
>
> MySQL可以为整数类型指定宽度，例如int(11)，对大多数应用这是没有意义的：它不会限制值的合法范围，只是规定了mysql的一些交互工具（例如mysql命令行客户端）用来显示字符的个数。对于存储和计算来说，int(1)和int(20)是相同的。

字符类型

VARCHAR和CHAR是两种最主要的字符串类型。

VARCHAR:

> varchar类型选用于存储可变长字符串，需要使用1或者2个额外字节记录字符串的长度，比定长类型更节省空间，因为它仅使用必要的空间（例如越短的字符串使用越少的空间）。由于VARCHAR行是变长的，在UPDATE时可能使行变得比原来更长，这就导致需要做额外的工作。

varchar的使用场景：

> 字符串列的最大长度比平均长度大很多；列的更新很少，所以碎片不是问题；使用了像utf-8这样复杂的字符集，每个字符都使用不同的字节数进行存储。

CHAR:

> CHAR类型是定长的，MySQL总是根据定义的字符串长度分配足够的空间。char适合于存储很短的字符串，或者所有值都接近同一个长度。例如char非常适合存储密码MD5值，因为这是一个定长的值。对于经常变更的数据，char也比varchar更好，因为定长的char类型不容易产生碎片。

大字段字符类型

> blob和text都是为了存储很大的数据而设计的字符串数据类型，分别采用二进制和字符串方式存储。MySQL对blob和text列进行排序于其他类型是不同的：它只对每个列的最前max_sort_length字节而不是整个字符串排序。
>
> 尽量避免使用text和blob类型，如果实在无法避免，有一个技巧实在所有用到blob字段的地方都是用substring(column,length)将列值转换为字符串（在order by 子句中也适用），这样就可以使用内存临时表了。
>
> 但是要确保截取的子字符串足够短，不会使临时表的大小超过max_heap_table_size或tmp_table_size，超过以后Mysql会将内存临时表转换为MyISAM磁盘临时表。

时间类型

> timestamp类型保存了从1970年1月1日午夜以来的秒数，timestamp只使用4个字节的存储空间，因此它的范围比>
>
> datatime小得多：只能表示从1970年到2038年。提供了from_unixtime函数把unix时间戳转换为日期、>
>
> unix_timestamp函数把日期转换为时间戳。 有时候人们会将unix时间戳存储为整数值，这不会带来任何收益。用整数保存时间戳的格式通常不方便处理，所以我们不推荐这么做。

标识符（identifier）

> 整数通常是标识列最好的选择，因为它们很快并且可以使用auto_increment；千万不要使用enum和set类型作为标识列；尽量避免使用字符串类型作为标识列，因为他们很消耗空间，并且通常比数字类型慢。尤其是在MyISAM表里使用字符串作为标识列时要特别小心，因为MyISAM默认对字符串使用压缩索引，这会导致查询慢得多。

特殊类型数据：IP地址字段（IPv4）

> 人们经常使用varchar(15)来存储ip地址，然而，它们实际上是32位无符号整数，不是字符串。MySQL提供INET_ATON()和INET_NTOA()函数将ip地址在整数和四段表示形式之间进行转换。

3.数据库中的范式和反范式

在范式化的数据库中，每个事实数据会出现并且只出现一次；相反，在反范式化的数据库中，信息是冗余的，可能会存储在多个地方。
范式化的优点：
1）范式化的更新操作通常比反范式化要快。
2）当数据较好地范式化时，就只有很少或者没有重复数据，所以只需要修改更少的数据。
3）范式化的表通常更小，可以更好地放在内存里，所以执行操作会更快。
4）很少有多余的数据意味着检索列表数据时更少需要DISTINCT或者GROUP BY语句。

范式化设计的schema的缺点是通常需要关联，较多的关联可能使得一些索引策略无效，例如，范式化可能将列存放在不同的表中，而这些列如果在一个表中本可以属于同一个索引。

> 反范式的schema因为所有数据都在一张表中，可以很好地避免关联。缺点是update操作的代价高，需要更新多个表，至于这会不会是一个问题，需要考虑更新的频率以及更新的时长，并和执行select查询的频率进行比较。
>
> 从另一个父表冗余一些数据到子表的理由通常是排序的需要。
>
> 缓存衍生值也是有用的。如果需要显示每个用户发了多少消息（像很多论坛做的），可以每次执行一个昂贵的子查询来计算并显示它；也可以在user表中建一个num_messages列，每当用户发新消息时更新这个值。

4.缓存表和汇总表

> s缓存表和汇总表，实时计算统计值是非常昂贵的操作，因为要么需要扫描表中的大部分数据，要么查询语句只能在某些特定的索引上才能有效运行，而这类特定索引一般会对update操作有影响，所以一般不希望创建这样的索引。
>
> 使用缓存表和汇总表时，必须决定是实时维护数据还是定期重建，哪个更好依赖于应用程序，但是定期重建并不只是节省资源，也可以保持表不会有很多碎片，以及有完全顺序组织的索引（这会更加高效）。

5.物化视图

> 物化视图实际上是预先计算并且存储在磁盘上的表，可以通过各种各样的策略刷新和更新。MySQL并不原生支持物化视图。使用开源工具Flexviews可以自己实现物化视图。它由下面这些部分组成：

1）变更数据抓取功能，可以读取服务器的二进制日志并且解析相关行的变更。
2）一系列可以帮助创建和管理视图的定义的存储过程
3）一些可以应用变更到数据库中的物化视图工具

6.计数器表

> 如果应用在表中保存计数器，在更新计数器时可能会碰到并发问题。有一个技巧：将计数器保存在多行中，更新计数+1的操作改为随机选择一行进行更新，求计数值的时候，做一个sum求和。

7.加快ALTER TABLE操作的速度（表结构更改）

> 一般而言，大部分alter table操作将导致MySQL服务中断（锁表并重建表）。MySQL执行大部分修改表结构的操作方法是用新的结构创建一个空表，从旧表中查出所有数据插入新表，然后删除旧表。这个操作可能需要花费很长时间。
>
> 不是所有的alter table操作都会引起表重建。理论上，MySQL可以跳过创建新表的步骤。列的默认值实际上存在表的.frm文件中，所以可以直接修改这个文件而不需要改动表本身。
>
> 比如使用 alter comlum改变列的默认值：

```
alert table tablename
alert column col1 set default 5;
```

这个语句会直接修改.frm文件而不涉及表数据，所以这个操作是非常快的。