# 目标

* 了解什么是优化
* 掌握优化查询的方法
* 掌握优化数据库结构的方法
* 掌握优化MySQL服务器的方法

# 什么是优化？

* 合理安排资源、调整系统参数使MySQL运行更快、更节省资源。
* 优化是多方面的，包括查询、更新、服务器等。
* 原则：减少系统瓶颈，减少资源占用，增加系统的反应速度。

IO次数

CPU的运算复杂度

# 数据库性能参数

* 使用SHOW STATUS语句查看MySQL数据库的性能参数
  + SHOW STATUS LIKE 'value‘
* 常用的参数：
  + Slow\_queries 慢查询次数
  + Com\_(CRUD) 操作的次数
  + Uptime 上线时间

# 查询优化

SQLA：复杂，IO 1000 次 频率10次

SQLB：IO 20次 10000次

**原则：优化更需要优化的SQL**

哪些Sql更需要优化？

执行频率更高的

定义Sql的性能瓶颈

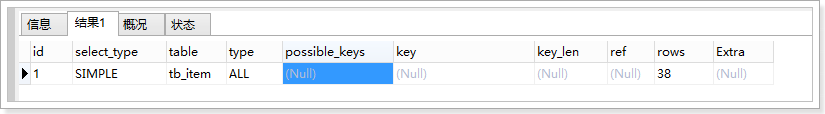
是IO引起的问题，还是CPU？是数据访问引起还是数据运算引起？

从哪里入手开始？

执行计划

## EXPLAIN

在MySQL中可以使用EXPLAIN查看SQL执行计划，用法：EXPLAIN SELECT \* FROM tb\_item



## 结果说明

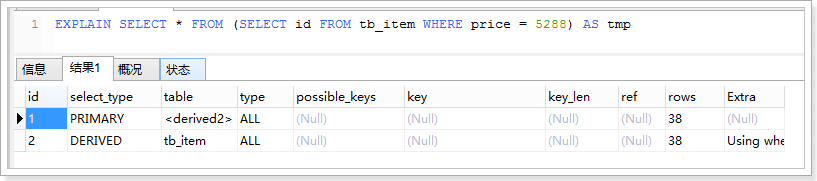
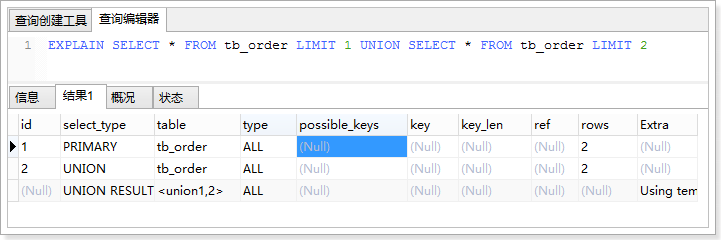
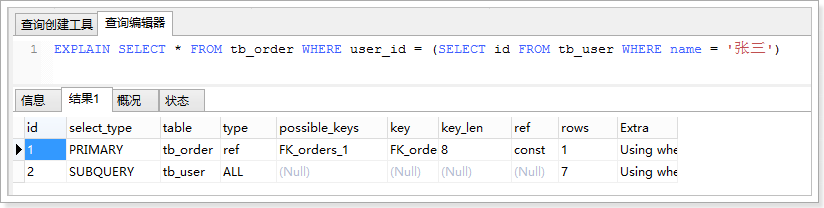
### id

SELECT识别符。这是SELECT查询序列号。这个不重要。

### select\_type

表示SELECT语句的类型。

有以下几种值：

1. SIMPLE  
   表示简单查询，其中不包含连接查询和子查询。
2. PRIMARY  
   表示主查询，或者是最外面的查询语句。  
   
3. UNION  
   表示连接查询的第2个或后面的查询语句。  
   
4. DEPENDENT UNION  
   UNION中的第二个或后面的SELECT语句，取决于外面的查询。
5. UNION RESULT  
   连接查询的结果。
6. SUBQUERY  
   子查询中的第1个SELECT语句。  
   
7. DEPENDENT SUBQUERY  
   子查询中的第1个SELECT语句，取决于外面的查询。
8. DERIVED  
   SELECT(FROM 子句的子查询)。

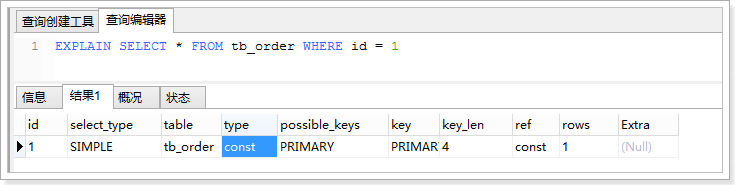
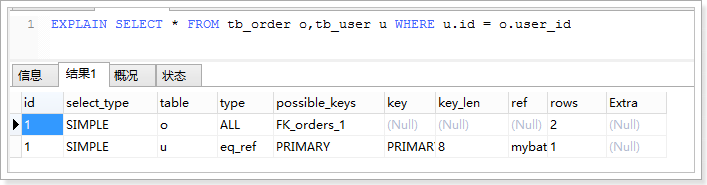
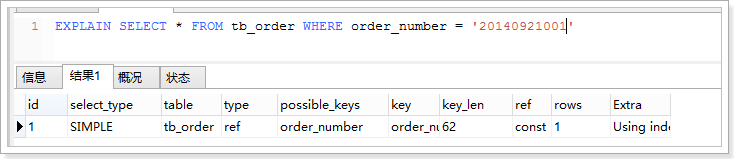
### table

表示查询的表。

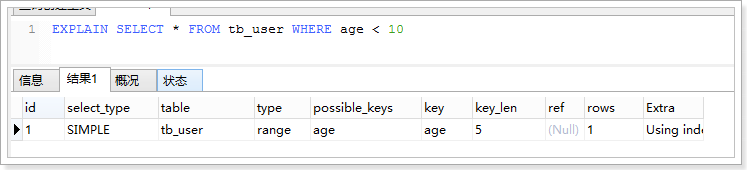
### type（重要）

表示表的连接类型。

以下的连接类型的顺序是从最佳类型到最差类型：

1. system  
   表仅有一行，这是const类型的特列，平时不会出现，这个也可以忽略不计。
2. const  
   数据表最多只有一个匹配行，因为只匹配一行数据，所以很快，常用于PRIMARY KEY或者UNIQUE索引的查询，可理解为const是最优化的。  
   
3. eq\_ref  
   mysql手册是这样说的:"对于每个来自于前面的表的行组合，从该表中读取一行。这可能是最好的联接类型，除了const类型。它用在一个索引的所有部分被联接使用并且索引是UNIQUE或PRIMARY KEY"。eq\_ref可以用于使用=比较带索引的列。  
   
4. ref  
   查询条件索引既不是UNIQUE也不是PRIMARY KEY的情况。ref可用于=或<或>操作符的带索引的列。  
   
5. ref\_or\_null  
   该联接类型如同ref，但是添加了MySQL可以专门搜索包含NULL值的行。在解决子查询中经常使用该联接类型的优化。

上面这五种情况都是很理想的索引使用情况。

1. index\_merge  
   该联接类型表示使用了索引合并优化方法。在这种情况下，key列包含了使用的索引的清单，key\_len包含了使用的索引的最长的关键元素。
2. unique\_subquery  
   该类型替换了下面形式的IN子查询的ref: value IN (SELECT primary\_key FROM single\_table WHERE some\_expr)   
   unique\_subquery是一个索引查找函数,可以完全替换子查询,效率更高。
3. index\_subquery  
   该联接类型类似于unique\_subquery。可以替换IN子查询,但只适合下列形式的子查询中的非唯一索引: value IN (SELECT key\_column FROM single\_table WHERE some\_expr)
4. range  
   只检索给定范围的行,使用一个索引来选择行。  
   
5. index  
   该联接类型与ALL相同,除了只有索引树被扫描。这通常比ALL快,因为索引文件通常比数据文件小。
6. ALL  
   对于每个来自于先前的表的行组合,进行完整的表扫描。（性能最差）

### possible\_keys

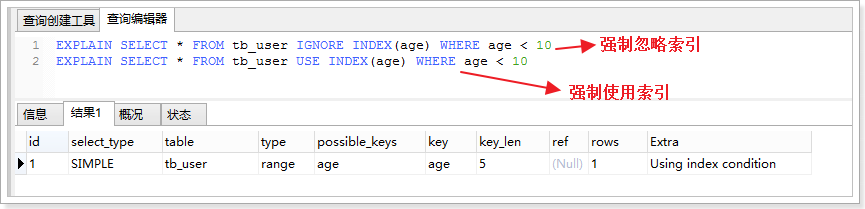
指出MySQL能使用哪个索引在该表中找到行。

如果该列为NULL，说明没有使用索引，可以对该列创建索引来提高性能。

### key

显示MySQL实际决定使用的键(索引)。如果没有选择索引,键是NULL。

可以强制使用索引或者忽略索引：



### key\_len

显示MySQL决定使用的键长度。如果键是NULL,则长度为NULL。

注意：key\_len是确定了MySQL将实际使用的索引长度。

前缀索引

### ref

显示使用哪个列或常数与key一起从表中选择行。

### rows

显示MySQL认为它执行查询时必须检查的行数。

### Extra

该列包含MySQL解决查询的详细信息

* Distinct:MySQL发现第1个匹配行后,停止为当前的行组合搜索更多的行。
* Not exists:MySQL能够对查询进行LEFT JOIN优化,发现1个匹配LEFT JOIN标准的行后,不再为前面的的行组合在该表内检查更多的行。
* range checked for each record (index map: #):MySQL没有发现好的可以使用的索引,但发现如果来自前面的表的列值已知,可能部分索引可以使用。
* Using filesort:MySQL需要额外的一次传递,以找出如何按排序顺序检索行。
* Using index:从只使用索引树中的信息而不需要进一步搜索读取实际的行来检索表中的列信息。
* Using temporary:为了解决查询,MySQL需要创建一个临时表来容纳结果。
* Using where:WHERE 子句用于限制哪一个行匹配下一个表或发送到客户。
* Using sort\_union(...), Using union(...), Using intersect(...):这些函数说明如何为index\_merge联接类型合并索引扫描。
* Using index for group-by:类似于访问表的Using index方式,Using index for group-by表示MySQL发现了一个索引,可以用来查 询GROUP BY或DISTINCT查询的所有列,而不要额外搜索硬盘访问实际的表。

## 查询优化注意事项

添加索引的条件：

1）经常作为查询条件（排序）的字段

2）不能经常修改

索引可以提供查询的速度，但并不是使用了带有索引的字段查询都会生效，有些情况下是不生效的，需要注意！

### 不在数据库做计算

禁止使用存储过程、视图、触发器、Event。

在并发量大的情况下，这些功能很可能将数据库拖跨，业务逻辑放到服务层具备更好的扩展性，能够轻易实现“增机器就加性能”

### 不同类型不可进行比较

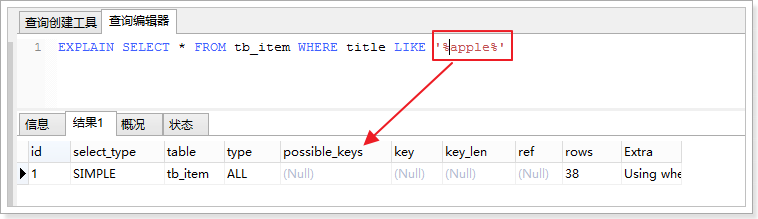
如果将不同类型比较，Mysql底层会自动进行数据类型转换，大大影响效率

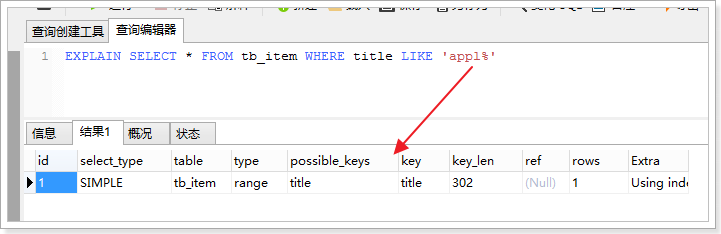
### 禁止使用反向查询

负向查询条件：NOT、!=、<>、!<、!>、NOT IN、NOT LIKE等，会导致全表扫描

### 尽量不使用LIKE关键字的查询

在使用LIKE关键字进行查询的查询语句中，如果匹配字符串的第一个字符为“%”，索引不起作用。只有“%”不在第一个位置，索引才会生效。

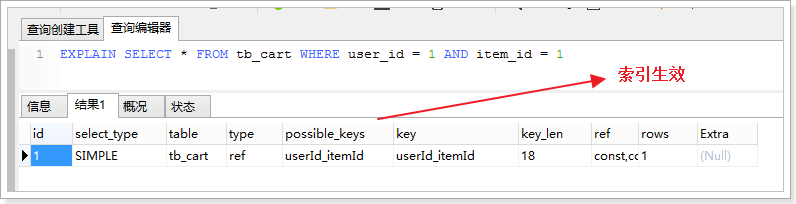


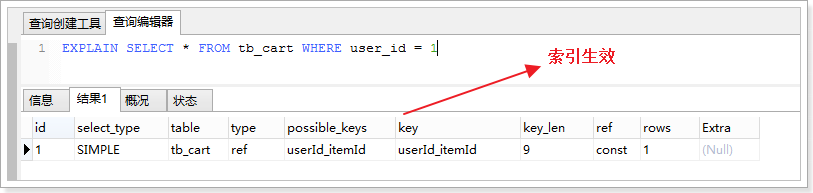


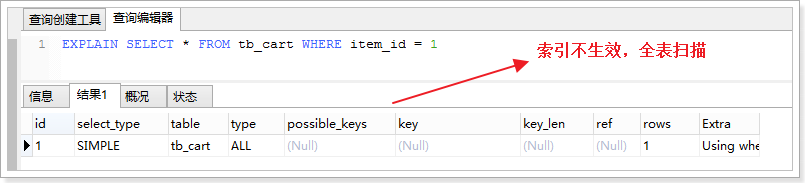
### 联合索引的查询的注意事项

MySQL可以为多个字段创建索引，一个索引可以包括16个字段。对于联合索引，只有查询条件中使用了这些字段中第一个字段时，索引才会生效。

A,B

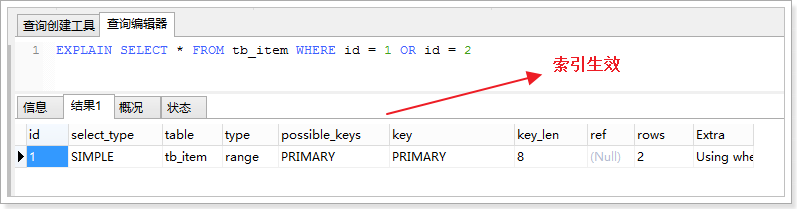


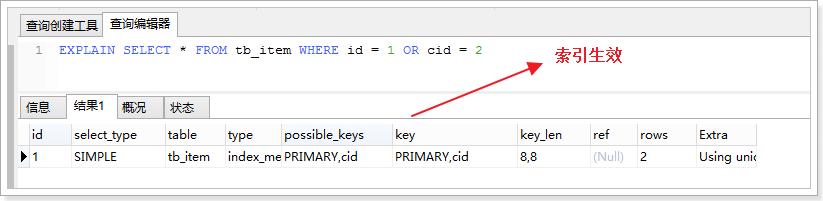


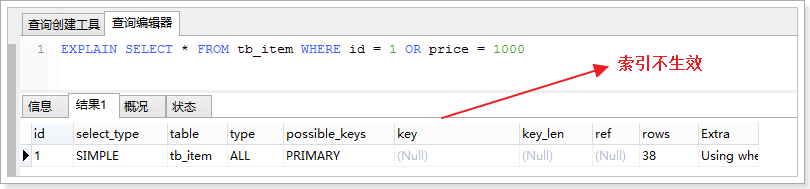


### 使用OR关键字的查询

查询语句的查询条件中只有OR关键字，且OR前后的两个条件中的列都是索引时，索引才会生效，否则，索引不生效。







### 尽量不使用OR关键字

在使用or时，如果条件中有一个不是索引字段，就会导致整个索引失效。

此时，推荐大家使用in 或者 union

### 查询条件中尽量不要使用函数

在条件中使用mysql函数，可能导致索引失效。

大多数的MySQL服务器都开启了查询缓存。这是提高性最有效的方法之一，而且这是被MySQL的数据库引擎处理的。当有很多相同的查询被执行了多次的时候，这些查询结果会被放到一个缓存中，这样，后续的相同的查询就不用操作表而直接访问缓存结果了。

这里最主要的问题是，对于程序员来说，这个事情是很容易被忽略的。因为，我们某些查询语句会让MySQL不使用缓存。请看下面的示例：

// 查询缓存不开启

SELECT username FROM user WHERE signup\_date >= CURDATE()

// 开启查询缓存

SELECT username FROM user WHERE signup\_date >= 2015-01-12

上面两条SQL语句的差别就是 CURDATE() ，MySQL的查询缓存对这个函数不起作用。所以，像 NOW() 和 RAND() 或是其它的诸如此类的SQL函数都不会开启查询缓存，因为这些函数的返回是会不定的易变的。所以，你所需要的就是用一个变量来代替MySQL的函数，从而 开启缓存。

### 当只要一行数据时使用 LIMIT 1

当你查询表的有些时候，你已经知道结果只会有一条结果，但因为你可能需要去fetch游标，或是你也许会去检查返回的记录数。

在这种情况下，加上 LIMIT 1 可以增加性能。这样一样，MySQL数据库引擎会在找到一条数据后停止搜索，而不是继续往后查少下一条符合记录的数据。

### 关联查询时，永远用小结果集驱动大的结果集

其实这样的结果也非常容易理解，在 MySQL 中的 Join，只有 Nested Loop 一种 Join 方式，也就是MySQL 的 Join 都是通过嵌套循环来实现的。驱动结果集越大，所需要循环的此时就越多，那么被驱动表的访问次数自然也就越多，而每次访问被驱动表，即使需要的逻辑 IO 很少，循环次数多了，总量自然也不可能很小，而且每次循环都不能避免的需要消耗 CPU ，所以 CPU 运算量也会跟着增加。

反之，所需要的循环次数就会更少，总体 IO 量和 CPU 运算量也会少。而且，就算是非 Nested Loop 的 Join 算法，如 Oracle 中的 Hash Join，同样是小结果集驱动大的结果集是最优的选择。

### 只取出自己需要的 Columns；不要select \*

任何时候在 Query 中都只取出自己需要的 Columns，尤其是在需要排序的 Query 中。

对于任何 Query，返回的数据都是需要通过网络数据包传回给客户端，如果取出的 Column 越多，需要传输的数据量自然会越大，不论是从网络带宽方面考虑还是从网络传输的缓冲区来看，都是一个浪费。

如果是需要排序的 Query 来说，影响就更大了。在 MySQL 中存在两种排序算法：

一种是在MySQL4.1 之前的老算法，实现方式是先将需要排序的字段和可以直接定位到相关行数据的指针信息取出，然后在我们所设定的排序区（通过参数 sort\_buffer\_size 设定）中进行排序，完成排序之后再次通过行指针信息取出所需要的 Columns，也就是说这种算法需要访问两次数据。

第二种排序算法是从MySQL4.1 版本开始使用的改进算法，一次性将所需要的 Columns 全部取出，在排序区中进行排序后直接将数据返回给请求客户端。改行算法只需要访问一次数据，减少了大量的随机IO，极大的提高了带有排序的 Query 语句的效率。但是，这种改进后的排序算法需要一次性取出并缓存的数据比第一种算法要多很多，如果我们将并不需要的 Columns 也取出来，就会极大的浪费排序过程所需要的内存。

### 尽量不使用子查询

MySQL从4.1版本开始支持子查询，使用子查询进行SELECT语句嵌套查询，可以一次完成很多逻辑上需要多个步骤才能完成的SQL操作。

子查询虽然很灵活，但是执行效率并不高。

执行子查询时，MYSQL需要创建临时表，查询完毕后再删除这些临时表，所以，子查询的速度会受到一定的影响。

优化：

可以使用连接查询（JOIN）代替子查询，连接查询时不需要建立临时表，其速度比子查询快。另外，join关联的条件，最好也是索引字段！

# 数据库结构优化

一个好的数据库设计方案对于数据库的性能往往会起到事半功倍的效果。

需要考虑数据冗余、查询和更新的速度、字段的数据类型是否合理等多方面的内容。

## 选择合适的数据类型（\*\*\*\*\*）

优化数据类型提高性能的主要原理在于以下几个方面：

1. 通过选用更“小”的数据类型减少存储空间，使查询相同数据需要的 IO 资源降低；

2. 通过合适的数据类型加速数据的比较

数字：

空间越小越好

能用整数就不要用小数

字符串：

能用数字就用数字。（用字典方式）

长度课固定就不固定

日期类型：

用数字（毫秒值）

Timestamp：存的是毫秒值

1970~

## 不要使用TEXT、BLOB类型

会浪费更多的磁盘和内存空间，非必要的大量的大字段查询会淘汰掉热数据，导致内存命中率急剧降低，影响数据库性能,如果必须要使用则独立出来一张表，用主键来对应，避免影响其它字段索引效率

## 把字段定义为NOT NULL并且提默认值

a）null的列使索引/索引统计/值比较都更加复杂，对MySQL来说更难优化

b）null 这种类型MySQL内部需要进行特殊处理，增加数据库处理记录的复杂性；同等条件下，表中有较多空字段的时候，数据库的处理性能会降低很多

c）null值需要更多的存储空间，无论是表还是索引中每行中的null的列都需要额外的空间来标识

d）对null 的处理时候，只能采用is null或is not null，而不能采用=、in、<、<>、!=、not in这些操作符号。如：where name!=’zhangsan’，如果存在name为null值的记录，查询结果就不会包含name为null值的记录

## 将字段很多的表分解成多个表

一般一张表字段不能超过50

对于字段较多的表，如果有些字段的使用频率很低，可以将这些字段分离出来形成新表。

因为当一个表的数据量很大时，会由于使用频率低的字段的存在而变慢。

垂直分表：

字段过多，按列分表

水平分表（慎用，会导致逻辑变复杂）：

表内数据过多，超过1000万，需要水平分表：按行分。mycat自动分库分表

## 增加中间表

对于需要经常联合查询的表，可以建立中间表以提高查询效率。

通过建立中间表，将需要通过联合查询的数据插入到中间表中，然后将原来的联合查询改为对中间表的查询。

A、B、C

A.x,A.y,A.z B.x C.y

D(A.x,A.y,A.z B.x C.y)

## 适当增加冗余字段

A(x,y,z, B.x,C.y)

设计数据表时应尽量遵循范式理论的规约，尽可能的减少冗余字段，让数据库设计看起来精致、优雅。但是，合理的加入冗余字段可以提高查询速度。

表的规范化程度越高，表和表之间的关系越多，需要连接查询的情况也就越多，性能也就越差。

**注意：**

**冗余字段的值在一个表中修改了，就要想办法在其他表中更新，否则就会导致数据不一致的问题。**

## 尽量不要设计表关联字段（外键）

外键会导致表与表之间耦合，update与delete操作都会涉及相关联的表，十分影响sql 的性能，甚至会造成死锁。高并发情况下容易造成数据库性能下降，大数据高并发业务场景数据库使用以性能优先

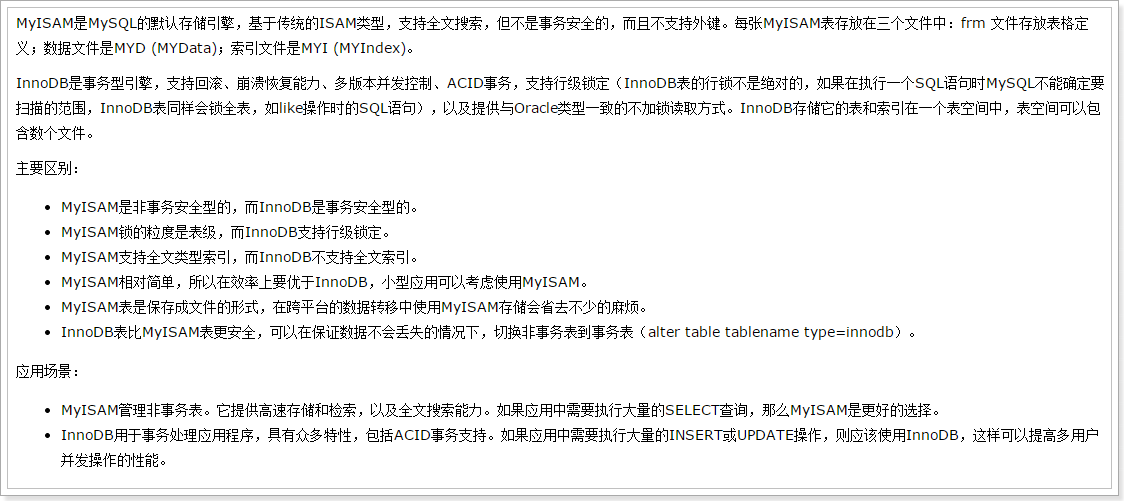
## 超过千万数据就要进行分表

# 插入数据的优化

插入数据时，影响插入速度的主要是索引、唯一性校验、一次插入的数据条数等。

插入数据的优化，不同的存储引擎优化手段不一样，在MySQL中常用的存储引擎有，MyISAM和InnoDB，两者的区别：

http://www.cnblogs.com/panfeng412/archive/2011/08/16/2140364.html



## MyISAM

### 禁用索引

对于非空表，插入记录时，MySQL会根据表的索引对插入的记录建立索引。如果插入大量数据，建立索引会降低插入数据速度。

为了解决这个问题，可以在批量插入数据之前禁用索引，数据插入完成后再开启索引。

禁用索引的语句：

*ALTER TABLE table\_name DISABLE KEYS*

开启索引语句：

*ALTER TABLE table\_name ENABLE KEYS*

对于空表批量插入数据，则不需要进行操作，因为MyISAM引擎的表是在导入数据后才建立索引。

### 禁用唯一性检查

唯一性校验会降低插入记录的速度，可以在插入记录之前禁用唯一性检查，插入数据完成后再开启。

禁用唯一性检查的语句：*SET UNIQUE\_CHECKS = 0;*

开启唯一性检查的语句：SET UNIQUE\_CHECKS = 1;

### 批量插入数据

插入数据时，可以使用一条INSERT语句插入一条数据，也可以插入多条数据。





第二种方式的插入速度比第一种方式快。

### 使用LOAD DATA INFILE

当需要批量导入数据时，使用LOAD DATA INFILE语句比INSERT语句插入速度快很多。

## InnoDB

### 禁用唯一性检查

用法和MyISAM一样。

### 禁用外键检查

插入数据之前执行禁止对外键的检查，数据插入完成后再恢复，可以提供插入速度。

禁用：SET foreign\_key\_checks = 0;

开启：SET foreign\_key\_checks = 1;

### 禁止自动提交

插入数据之前执行禁止事务的自动提交，数据插入完成后再恢复，可以提高插入速度。

禁用：SET autocommit = 0;

开启：SET autocommit = 1;

# 服务器优化

## 优化服务器硬件

服务器的硬件性能直接决定着MySQL数据库的性能，硬件的性能瓶颈，直接决定MySQL数据库的运行速度和效率。

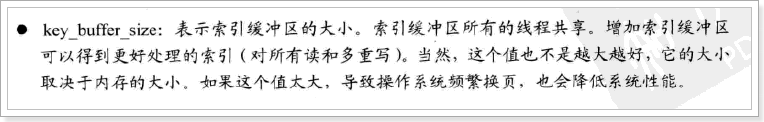
需要从以下几个方面考虑：

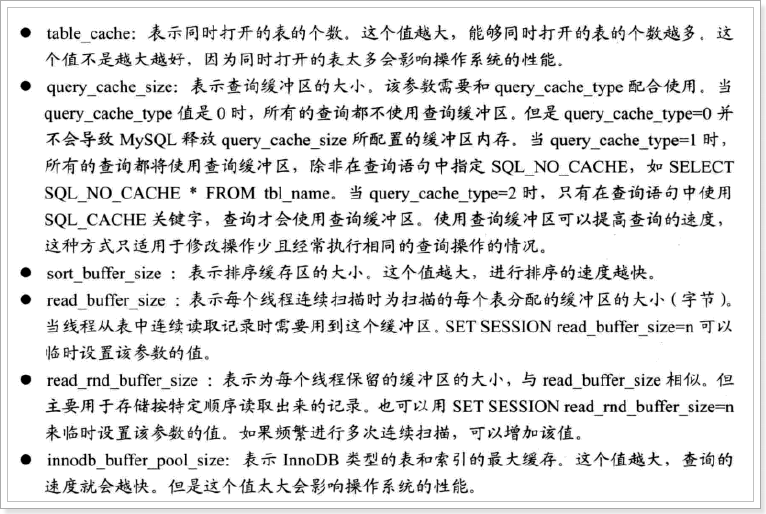
1. 配置较大的内存。足够大的内存，是提高MySQL数据库性能的方法之一。内存的IO比硬盘快的多，可以增加系统的缓冲区容量，使数据在内存停留的时间更长，以减少磁盘的IO。
2. 配置高速磁盘，比如SSD。
3. 合理分配磁盘IO，把磁盘IO分散到多个设备上，以减少资源的竞争，提高并行操作能力。
4. 配置多核处理器，MySQL是多线程的数据库，多处理器可以提高同时执行多个线程的能力。

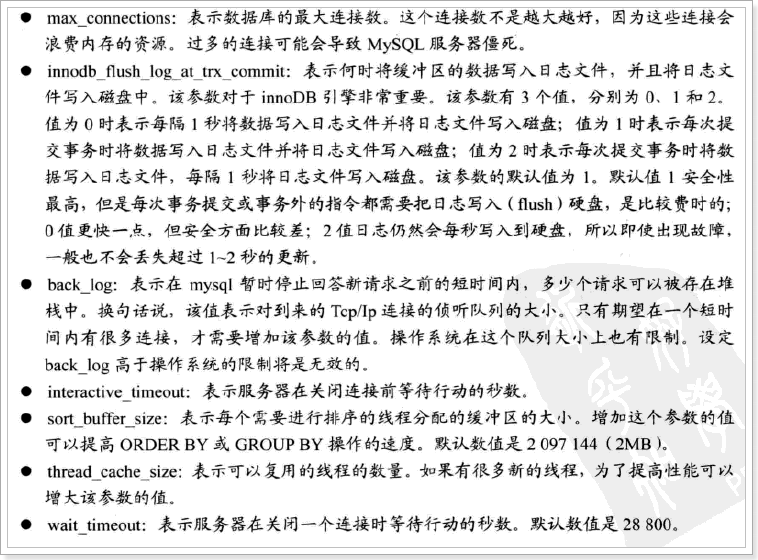
## 优化MySQL的参数

通过优化MySQL的参数可以提高资源利用率，从而达到提高MySQL服务器性能的目的。

MySQL的配置参数都在my.conf或者my.ini文件的[mysqld]组中，常用的参数如下：







要求：必须记忆至少3个。