目录

[一、公共 4](#_Toc15976611)

[索引 4](#_Toc15976612)

[共用SQL(起码两种数据库相同) 10](#_Toc15976613)

[(1)查询重复数据 10](#_Toc15976614)

[(2)查询分组数量 10](#_Toc15976615)

[(3)两列数据互换 11](#_Toc15976616)

[不加inner/left join 11](#_Toc15976617)

[二、MySQL 12](#_Toc15976618)

[MySQL命令行常用操作 12](#_Toc15976619)

[登录 12](#_Toc15976620)

[创建数据库 14](#_Toc15976621)

[MySQL如何执行查询 15](#_Toc15976622)

[开启Binlog日志 22](#_Toc15976623)

[数据表存储的位置 24](#_Toc15976624)

[远程访问权限的问题 24](#_Toc15976625)

[中文乱码的问题 25](#_Toc15976626)

[导入SQL文件提示” [Err] 2013 - Lost connection to MySQL server during query”的问题 25](#_Toc15976627)

[windows下区分大小写的问题 26](#_Toc15976628)

[mysql探究之null与not null 27](#_Toc15976629)

[慢查询 29](#_Toc15976630)

[命令行里访问mysql库 32](#_Toc15976631)

[mysql重启 33](#_Toc15976632)

[修改mysql密码 33](#_Toc15976633)

[远程登录mysql 33](#_Toc15976634)

[默认无法远程登录mysql的解决方法 34](#_Toc15976635)

[添加用户账号和权限 34](#_Toc15976636)

[mysql自动备份 35](#_Toc15976637)

[强制使用索引 36](#_Toc15976638)

[mysql性能优化的步骤 38](#_Toc15976639)

[死锁 41](#_Toc15976640)

[SQL查询需要注意的 41](#_Toc15976641)

[Text类型的字段（对性能影响） 42](#_Toc15976642)

[mysql性能优化- profile 42](#_Toc15976643)

[外键 44](#_Toc15976644)

[常用查询语句 45](#_Toc15976645)

[时间戳 45](#_Toc15976646)

[一对多组合查询 45](#_Toc15976647)

[LIMIT 45](#_Toc15976648)

[MySQL中怎么对varchar类型排序问题 49](#_Toc15976649)

[查询结果前加序号 49](#_Toc15976650)

[三、SQLLite 50](#_Toc15976651)

[四、SQLServer 50](#_Toc15976652)

[五、Oracle 50](#_Toc15976653)

[1、访问 50](#_Toc15976654)

[命令行方式 50](#_Toc15976655)

[Navicat 51](#_Toc15976656)

[2、常见错误 51](#_Toc15976657)

[3、查询语句和函数： 53](#_Toc15976658)

[4、PL/SQL： 54](#_Toc15976659)

[5、日志和维护： 56](#_Toc15976660)

[6、性能调优： 56](#_Toc15976661)

[7、Navicat： 56](#_Toc15976662)

[六、Redis 56](#_Toc15976663)

[1、进入redis 56](#_Toc15976664)

[2、远程连接redis 57](#_Toc15976665)

[3、设置redis密码 58](#_Toc15976666)

[4、Redis客户端的GUI工具 60](#_Toc15976667)

[99、redis和memcached比较 60](#_Toc15976668)

[九、大数据 60](#_Toc15976669)

[十、效率 61](#_Toc15976670)

# 一、公共

索引

索引主要有四类：

(1)主键索引

它是一种特殊的[唯一索引](https://www.baidu.com/s?wd=%E5%94%AF%E4%B8%80%E7%B4%A2%E5%BC%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)，不允许有空值。一般是在建表的时候同时创建主键索引

(2)普通索引

这是最基本的索引，它没有任何限制。

mysql支持前缀索引，一般姓名不会超过20个字符，所以我们这里建立索引的时候限定了长度20，这样可以节省[索引文件](https://www.baidu.com/s?wd=%E7%B4%A2%E5%BC%95%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)大小。

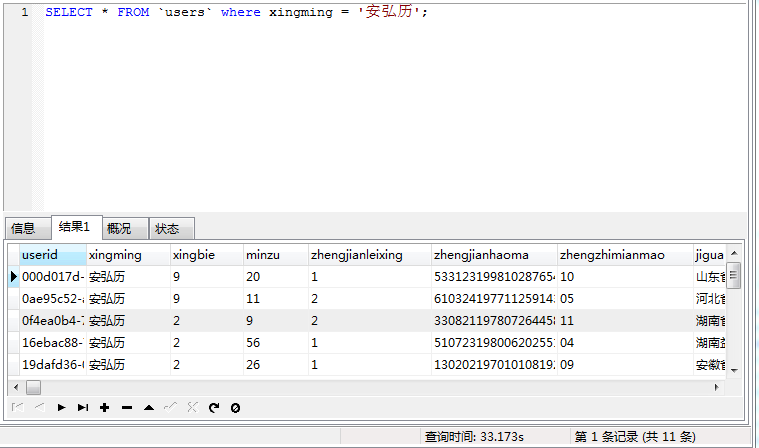
(3)[唯一索引](https://www.baidu.com/s?wd=%E5%94%AF%E4%B8%80%E7%B4%A2%E5%BC%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)

它与前面的普通索引类似，不同的就是：索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一。

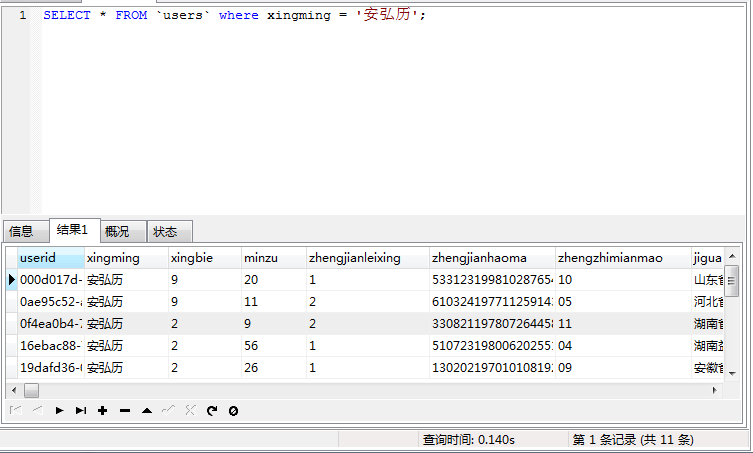
(4)[全文索引](https://www.baidu.com/s?wd=%E5%85%A8%E6%96%87%E7%B4%A2%E5%BC%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)

MySQL支持[全文索引](https://www.baidu.com/s?wd=%E5%85%A8%E6%96%87%E7%B4%A2%E5%BC%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)和搜索功能。MySQL中的[全文索引](https://www.baidu.com/s?wd=%E5%85%A8%E6%96%87%E7%B4%A2%E5%BC%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pj6dPWfdmH6smy7BnHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWbknjDLnWfznjnYPHc4rHcY)类型为FULLTEXT的索引。 FULLTEXT 索引仅可用于 MyISAM表。

使用索引前(150000+数据)：



使用索引后：



因此，对于数据库优化来说，建立有效的索引(太多的无用的索引也会影响效率)是很重要的。

通常情况下，索引会提高我们的查询效率，但是会降低我们的写入效率。因此索引也不是越多越好。而且由于数据库在查询的过程中也需要分析使用哪个索引，因此过多的索引会影响数据库分析的过程，从而影响查询的效率。

因此，有时我们还需要找出重复索引，比如主键就是一种唯一索引，因此如果你再增加一列唯一索引，对应的字段是主键id，这就会造成索引的重复创建。

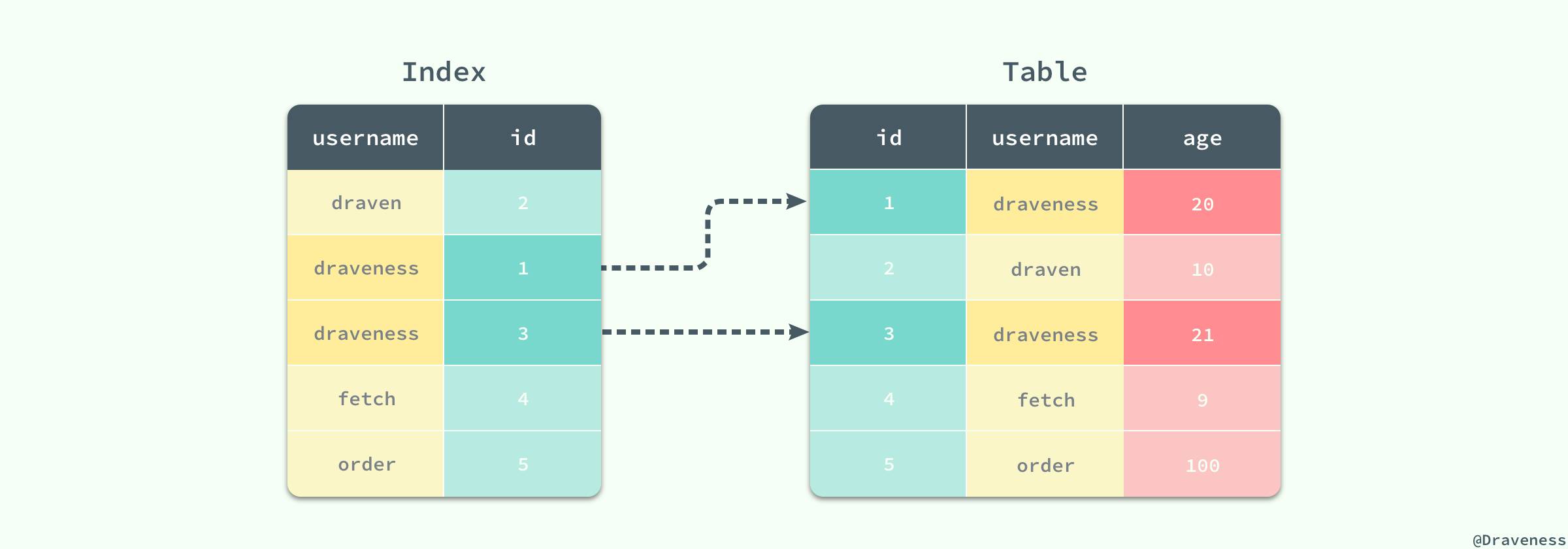
我们还需要知道数据库对数据的读取并不是以行为单位进行的，无论是读取一行还是多行，都会将该行或者多行所在的页全部加载进来，然后再读取对应的数据记录；也就是说，读取所耗费的时间与行数无关，只与页数有关。

数据库查询操作的时间大都消耗在从磁盘或者内存中读取数据的过程，由于随机 IO 的代价巨大，如何在一次数据库查询中减少随机 IO 的次数往往能够大幅度的降低查询所耗费的时间提高磁盘的吞吐量。

索引片其实就是 SQL 查询在执行过程中扫描的一个索引片段，在这个范围中的索引将被顺序扫描，根据索引片包含的列数不同，《[数据库索引设计与优化](https://www.amazon.cn/%E5%9B%BE%E4%B9%A6/dp/B00ZH27RH0)》 书中对将索引分为宽索引和窄索引：

对于查询 SELECT id, username, age FROM users WHERE username="draven" 来说，(id, username) 就是一个窄索引，因为该索引没有包含存在于 SQL 查询中的 age 列，而 (id, username, age) 就是该查询的一个宽索引了，它包含这个查询中所需要的全部数据列。

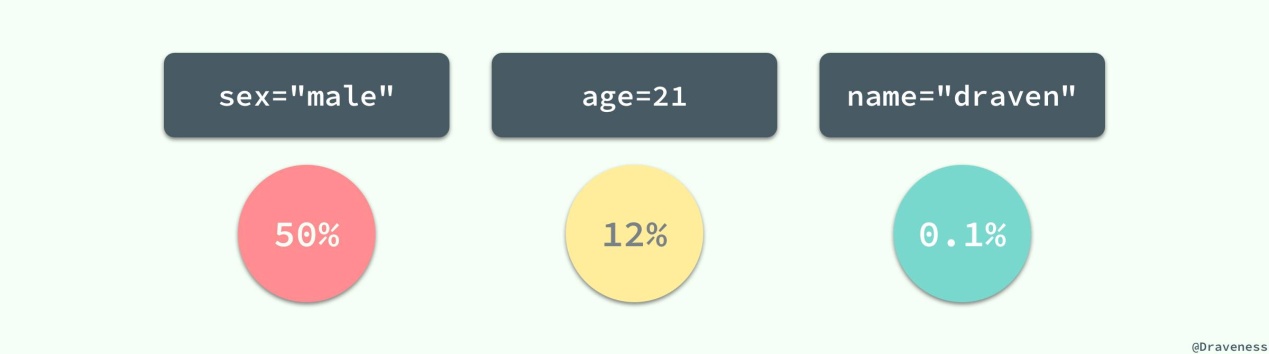
宽索引能够避免二次的随机 IO，而窄索引就需要在对索引进行顺序读取之后再根据主键 id 从主键索引中查找对应的数据。



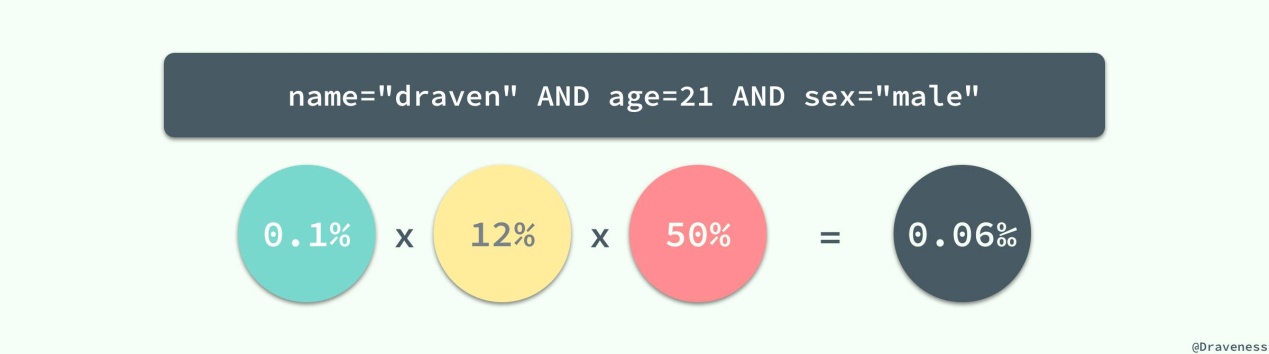
对于窄索引，每一个在索引中匹配到的记录行最终都需要执行另外的随机读取从聚集索引中获得剩余的数据，如果结果集非常大，那么就会导致随机读取的次数过多进而影响性能。

我们可以看到影响 SQL 查询的除了查询本身还与数据库表中的数据特征有关，如果使用的是窄索引那么对表的随机访问就不可避免，在这时如何让索引片变『薄』就是我们需要做的了。

一个 SQL 查询扫描的索引片大小其实是由过滤因子决定的，也就是满足查询条件的记录行数所占的比例：



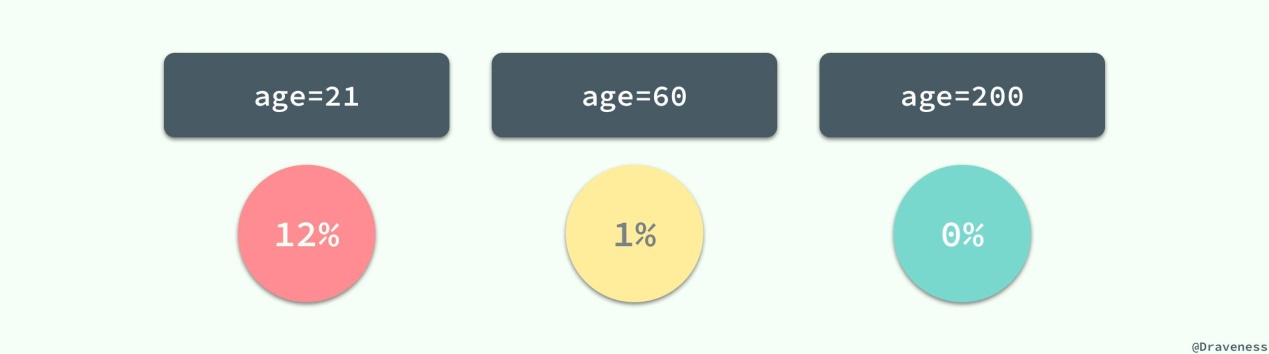
对于 users 表来说，sex=”male” 就不是一个好的过滤因子，它会选择整张表中一半的数据，所以**在一般情况下**我们最好不要使用 sex 列作为整个索引的第一列；而 name=”draven” 的使用就可以得到一个比较好的过滤因子了，它的使用能过滤整个数据表中 99.9% 的数据；当然我们也可以将这三个过滤进行组合，创建一个新的索引 (name, age, sex) 并同时使用这三列作为过滤条件：



当三个过滤条件都是等值谓词时，几个索引列的顺序其实是无所谓的，索引列的顺序不会影响同一个 SQL 语句对索引的选择，也就是索引 (name, age, sex) 和 (age, sex, name) 对于上图中的条件来说是完全一样的，这两个索引在执行查询时都有着完全相同的效果。

组合条件的过滤因子就可以达到十万分之6了，如果整张表中有 10w 行数据，也只需要在扫描薄索引片后进行 6 次随机读取，这种直接使用乘积来计算组合条件的过滤因子其实有一个比较重要的问题：列与列之间不应该有太强的相关性，如果不同的列之间有相关性，那么得到的结果就会比直接乘积得出的结果大一些，比如：所在的城市和邮政编码就有非常强的相关性，两者的过滤因子直接相乘其实与实际的过滤因子会有很大的偏差，不过这在多数情况下都不是太大的问题。

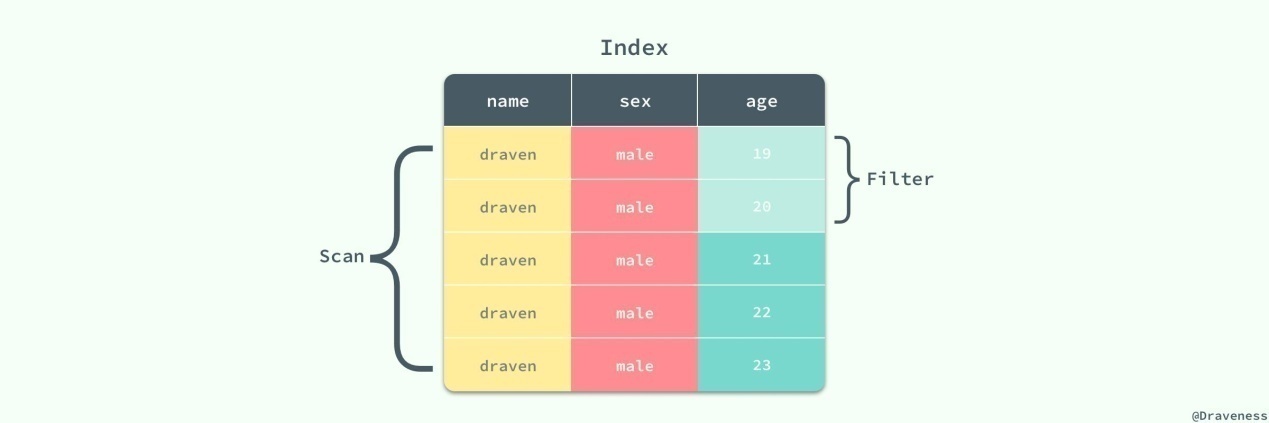
对于一张表中的同一个列，不同的值也会有不同的过滤因子，这也就造成了同一列的不同值最终的查询性能也会有很大差别：



假设在 users 表中有 name、age 和 (name, sex, age) 三个辅助索引；当 WHERE 条件中存在类似 age = 21 或者 name = “draven” 这种等值谓词时，它们都会成为匹配列（Matching Column）用于选择索引树中的数据行，但是当我们使用以下查询时：

SELECT \* FROM users WHERE name = "draven" AND sex = "male" AND age > 20;

虽然我们有 (name, sex, age) 索引包含了上述查询条件中的全部列，但是在这里只有 name 和 sex 两列才是匹配列，MySQL 在执行上述查询时，会选择 name 和 sex 作为匹配列，扫描所有满足条件的数据行，然后将 age 当做过滤列（Filtering Column）：



过滤列虽然不能够减少索引片的大小，但是能够减少从表中随机读取数据的次数，所以在索引中也扮演着非常重要的角色。

优化的原则（转载，日后逐一验证）：

(1)对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

(2)应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num is null

可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：select id from t where num=**0**

(3)应尽量避免在 where 子句中使用!=或<>操作符，否则引擎将放弃使用索引而进行全表扫描。

(4)应尽量避免在 where 子句中使用or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num=10 or num=20,可以这样查询：select id from t where num=10 union all select id from t where num=20。

(5)in和 not in也要慎用，否则会导致全表扫描，如：select id from t where num in(1,2,3) 对于连续的数值，能用 between 就不要用 in 了：select id from t where num between 1 and 3

(6)下面的查询也将导致全表扫描：select id from t where name like '李%'

共用SQL(起码两种数据库相同)

(1)查询重复数据

SELECT 项目ID,count(项目ID) as num FROM `商务评分标准` group by 项目ID HAVING num>1;

(2)查询分组数量

统计项目标段数量(mysql下可以这么写)：

SELECT xm.项目名称,count(bd.`标段ID`) as 标段数量 FROM `项目` xm INNER join `标段` bd on bd.`项目ID`=xm.`项目ID` group by xm.项目ID

统计一级学科包含的二级学科数量(oracle下必须这么写，用到的字段都包含在group by中):

select yjxk.一级学科名称,count(ejxk.二级学科id) as 二级学科数 from 一级学科 yjxk left join 二级学科 ejxk on yjxk.一级学科id=ejxk.一级学科id group by yjxk.一级学科id,yjxk.一级学科名称,yjxk.排序号 order by yjxk.排序号

(3)两列数据互换

MS SQL: update `ttsurvay\_model` t set t.carrieroperator=t.profession,t.profession=t.carrieroperator

但是MYSQL可以，否则两列相同了。

MYSQL: UPDATE ttsurvay\_model t1,ttsurvay\_model t2 SET t1.carrieroperator=t1.profession,t2.profession=t2.carrieroperator

不加inner/left join

比较了下列三个语句：

A．

SELECT COUNT(T.cityId) FROM `static\_form` t,

roles.t\_dic\_area c WHERE c.AREA\_ID=t.cityId

B．

SELECT COUNT(T.cityId) FROM `static\_form` t

INNER JOIN roles.t\_dic\_area c on c.AREA\_ID=T.cityId

C．

SELECT COUNT(T.cityId) FROM `static\_form` t

LEFT JOIN roles.t\_dic\_area c on c.AREA\_ID=T.cityId

发现A等同于B。

# 二、MySQL

MySQL命令行常用操作

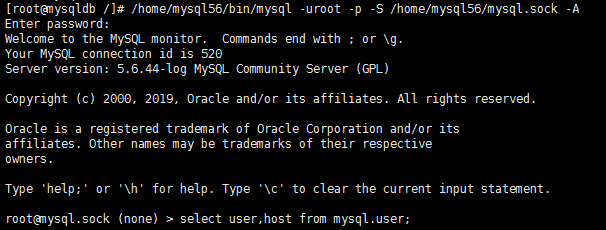
登录

mysql -uroot -p

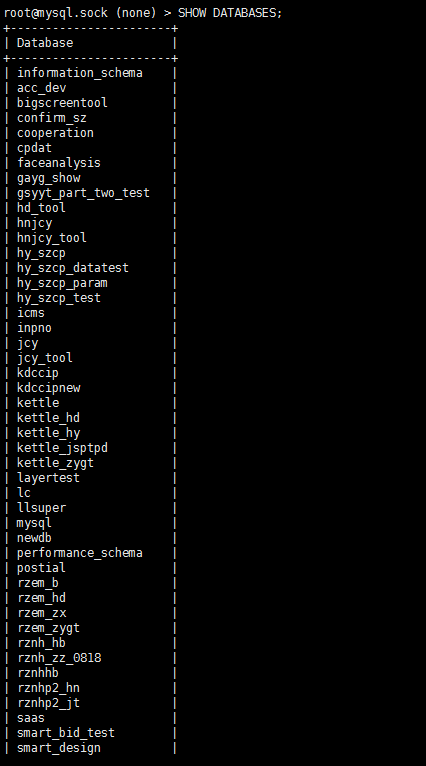


如果未找到命令，可以查看配置文件：





查看数据库：



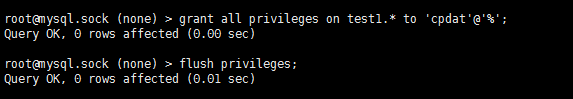
创建数据库



CREATE DATABASE IF NOT EXISTS test1 default character set utf8 COLLATE utf8\_general\_ci;

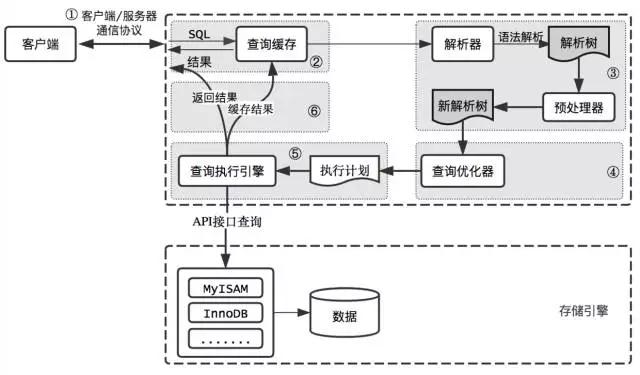
赋予cpdat用户的访问权限：

grant all privileges on test1.\* to 'cpdat'@'%';



MySQL如何执行查询

当向MySQL发送一个请求的时候，MySQL到底做了些什么呢？



客户端/服务端通信协议

MySQL客户端/服务端通信协议是“半双工”的：在任一时刻，要么是服务器向客户端发送数据，要么是客户端向服务器发送数据，这两个动作不能同时发生。一旦一端开始发送消息，另一端要接收完整个消息才能响应它，所以我们无法也无须将一个消息切成小块独立发送，也没有办法进行流量控制。

客户端用一个单独的数据包将查询请求发送给服务器，所以当查询语句很长的时候，需要设置max\_allowed\_packet参数。但是需要注意的是，如果查询实在是太大，服务端会拒绝接收更多数据并抛出异常。

与之相反的是，服务器响应给用户的数据通常会很多，由多个数据包组成。但是当服务器响应客户端请求时，客户端必须完整的接收整个返回结果，而不能简单的只取前面几条结果，然后让服务器停止发送。因而在实际开发中，尽量保持查询简单且只返回必需的数据，减小通信间数据包的大小和数量是一个非常好的习惯，这也是查询中尽量避免使用SELECT \*以及加上LIMIT限制的原因之一。

查询缓存

在解析一个查询语句前，如果查询缓存是打开的，那么MySQL会检查这个查询语句是否命中查询缓存中的数据。如果当前查询恰好命中查询缓存，在检查一次用户权限后直接返回缓存中的结果。这种情况下，查询不会被解析，也不会生成执行计划，更不会执行。

MySQL将缓存存放在一个引用表（不要理解成table，可以认为是类似于HashMap的数据结构），通过一个哈希值索引，两个查询在任何字符上的不同（例如：空格、注释），都会导致缓存不会命中。

如果查询中包含任何用户自定义函数、存储函数、用户变量、临时表、MySQL库中的系统表，其查询结果都不会被缓存。

既然是缓存，就会失效，那查询缓存何时失效呢？MySQL的查询缓存系统会跟踪查询中涉及的每个表，如果这些表（数据或结构）发生变化，那么和这张表相关的所有缓存数据都将失效。正因为如此，在任何的写操作时，MySQL必须将对应表的所有缓存都设置为失效。如果查询缓存非常大或者碎片很多，这个操作就可能带来很大的系统消耗，甚至导致系统僵死一会儿。而且查询缓存对系统的额外消耗也不仅仅在写操作，读操作也不例外：

1.任何的查询语句在开始之前都必须经过检查，即使这条SQL语句永远不会命中缓存。

2.如果查询结果可以被缓存，那么执行完成后，会将结果存入缓存，也会带来额外的系统消耗。

基于此，我们要知道并不是什么情况下查询缓存都会提高系统性能，缓存和失效都会带来额外消耗，只有当缓存带来的资源节约大于其本身消耗的资源时，才会给系统带来性能提升。

如果系统确实存在一些性能问题，可以尝试打开查询缓存，并在数据库设计上做一些优化，比如：

1.用多个小表代替一个大表，注意不要过度设计

2.批量插入代替循环单条插入

3.合理控制缓存空间大小，一般来说其大小设置为几十兆比较合适

可以通过SQL\_CACHE和SQL\_NO\_CACHE来控制某个查询语句是否需要进行缓存。

最后的忠告是不要轻易打开查询缓存，特别是写密集型应用。如果你实在是忍不住，可以将query\_cache\_type设置为DEMAND，这时只有加入SQL\_CACHE的查询才会走缓存，其他查询则不会，这样可以非常自由地控制哪些查询需要被缓存。

语法解析和预处理

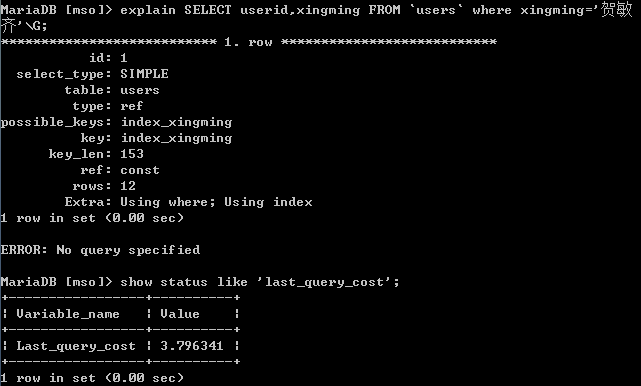
MySQL通过关键字将SQL语句进行解析，并生成一颗对应的解析树。这个过程解析器主要通过语法规则来验证和解析。比如SQL中是否使用了错误的关键字或者关键字的顺序是否正确等等。预处理则会根据MySQL规则进一步检查解析树是否合法。比如检查要查询的数据表和数据列是否存在等。

查询优化

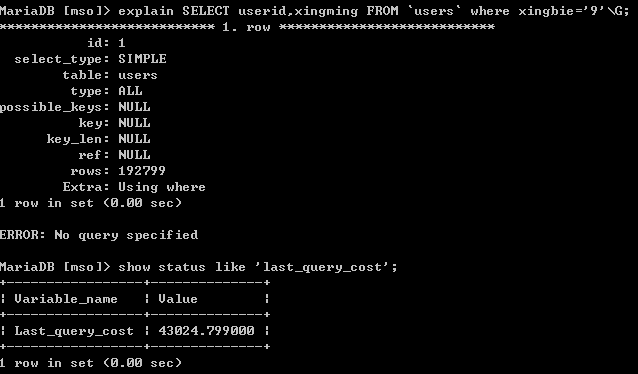
经过前面的步骤生成的语法树被认为是合法的了，并且由优化器将其转化成查询计划。多数情况下，一条查询可以有很多种执行方式，最后都返回相应的结果。优化器的作用就是找到这其中最好的执行计划。

MySQL使用基于成本的优化器，它尝试预测一个查询使用某种执行计划时的成本，并选择其中成本最小的一个。在MySQL可以通过查询当前会话的last\_query\_cost的值来得到其计算当前查询的成本。

使用索引时：



不使用索引时：



示例中的结果表示优化器认为大概需要做43024个数据页的随机查找才能完成上面的查询。这个结果是根据一些列的统计信息计算得来的，这些统计信息包括：每张表或者索引的页面个数、索引的基数、索引和数据行的长度、索引的分布情况等等。

查询执行引擎

在完成解析和优化阶段以后，MySQL会生成对应的执行计划，查询执行引擎根据执行计划给出的指令逐步执行得出结果。整个执行过程的大部分操作均是通过调用存储引擎实现的接口来完成，这些接口被称为handler API。查询过程中的每一张表由一个handler实例表示。实际上，MySQL在查询优化阶段就为每一张表创建了一个handler实例，优化器可以根据这些实例的接口来获取表的相关信息，包括表的所有列名、索引统计信息等。存储引擎接口提供了非常丰富的功能，但其底层仅有几十个接口，这些接口像搭积木一样完成了一次查询的大部分操作。

返回结果给客户端

查询执行的最后一个阶段就是将结果返回给客户端。即使查询不到数据，MySQL仍然会返回这个查询的相关信息，比如该查询影响到的行数以及执行时间等。

如果查询缓存被打开且这个查询可以被缓存，MySQL也会将结果存放到缓存中。

结果集返回客户端是一个增量且逐步返回的过程。有可能MySQL在生成第一条结果时，就开始向客户端逐步返回结果集了。这样服务端就无须存储太多结果而消耗过多内存，也可以让客户端第一时间获得返回结果。需要注意的是，结果集中的每一行都会以一个满足①中所描述的通信协议的数据包发送，再通过TCP协议进行传输，在传输过程中，可能对MySQL的数据包进行缓存然后批量发送。

回头总结一下MySQL整个查询执行过程，总的来说分为6个步骤：

a.客户端向MySQL服务器发送一条查询请求

b.服务器首先检查查询缓存，如果命中缓存，则立刻返回存储在缓存中的结果。否则进入下一阶段

c.服务器进行SQL解析、预处理、再由优化器生成对应的执行计划

d.MySQL根据执行计划，调用存储引擎的API来执行查询

e.将结果返回给客户端，同时缓存查询结果

性能优化建议

1.Scheme设计与数据类型优化

选择数据类型只要遵循小而简单的原则就好，越小的数据类型通常会更快，占用更少的磁盘、内存，处理时需要的CPU周期也更少。越简单的数据类型在计算时需要更少的CPU周期，比如，整型就比字符操作代价低，因而会使用整型来存储ip地址，使用DATETIME来存储时间，而不是使用字符串。

2.创建高性能索引

索引是提高MySQL查询性能的一个重要途径，但过多的索引可能会导致过高的磁盘使用率以及过高的内存占用，从而影响应用程序的整体性能。应当尽量避免事后才想起添加索引，因为事后可能需要监控大量的SQL才能定位到问题所在，而且添加索引的时间肯定是远大于初始添加索引所需要的时间，可见索引的添加也是非常有技术含量的。

3.索引相关的数据结构和算法

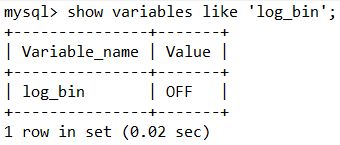
通常我们所说的索引是指B-Tree索引，它是目前关系型数据库中查找数据最为常用和有效的索引，大多数存储引擎都支持这种索引。

详见《万字总结：学习MySQL优化原理，这一篇就够了！(干货)》。

开启Binlog日志

Navicat下F6，打开对话框：

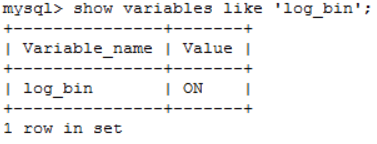
mysql> show variables like 'log\_bin';



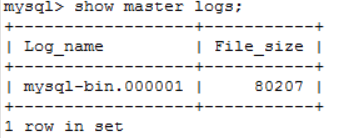
因此mysql查询某个参数：show variables like，如：show variables like 'log\_%';

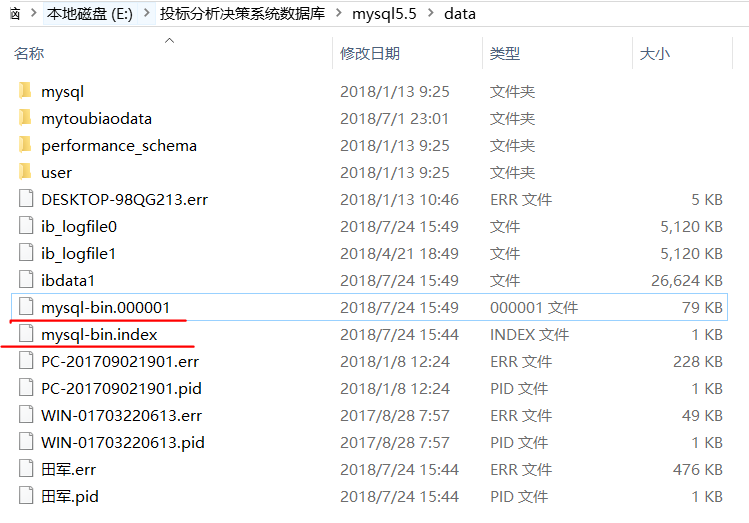
在my.ini（Linux下是my.cnf）中添加：

log-bin=mysql-bin



Binlog日志开启。





binlog是二进制文件，普通文件查看器cat more vi等都无法打开，必须使用自带的 mysqlbinlog 命令查看。

这里介绍一种较为方便的查询命令：

mysql> show binlog events [IN 'log\_name'] [FROM pos] [LIMIT [offset,] row\_count];

选项解析：

IN 'log\_name' 指定要查询的binlog文件名(不指定就是第一个binlog文件)

FROM pos 指定从哪个pos起始点开始查起(不指定就是从整个文件首个pos点开始算)

LIMIT [offset,] 偏移量(不指定就是0)

row\_count 查询总条数(不指定就是所有行)

这条语句可以将指定的binlog日志文件，分成有效事件行的方式返回，并可使用limit指定pos点的起始偏移，查询条数；

A.查询第一个(最早)的binlog日志：

mysql> show binlog events\G;

B.指定查询 mysql-bin.000021 这个文件：

mysql> show binlog events in 'mysql-bin.000021'\G;

C.指定查询 mysql-bin.000021 这个文件，从pos点:8224开始查起：

mysql> show binlog events in 'mysql-bin.000021' from 8224\G;

D.指定查询 mysql-bin.000021 这个文件，从pos点:8224开始查起，查询10条

mysql> show binlog events in 'mysql-bin.000021' from 8224 limit 10\G;

E.指定查询 mysql-bin.000021 这个文件，从pos点:8224开始查起，偏移2行，查询10条

mysql> show binlog events in 'mysql-bin.000021' from 8224 limit 2,10\G;

详情可见：<https://www.cnblogs.com/martinzhang/p/3454358.html>

数据表存储的位置

数据库保存在information\_schema库中，SCHEMATA表中。

远程访问权限的问题

登录数据库：

use mysql;

%为所有ip都可以远程访问：

update user set host=’%’ where user=’root’;

推送设置到内存或重启服务器也行：

flush privileges

mysql插入二进制文件报错：Packets larger than max\_allowed\_packet are not all

在导大容量数据特别是CLOB数据时，可能会出现异常：Packets larger than max\_allowed\_packet are not allowed。

这是由于MySQL数据库有一个系统参数max\_allowed\_packet，其默认值为1048576(1M)，可以通过如下语句在数据库中查询其值：

show VARIABLES like '%max\_allowed\_packet%';

修改此参数的方法是在mysql文件夹找到my.ini文件，在my.ini文件[mysqld]中添加一行：

max\_allowed\_packet=104857600

重启MySQL，这样将可以导入不大于100M的数据了，当然这数值可以根据需要作调整。

中文乱码的问题

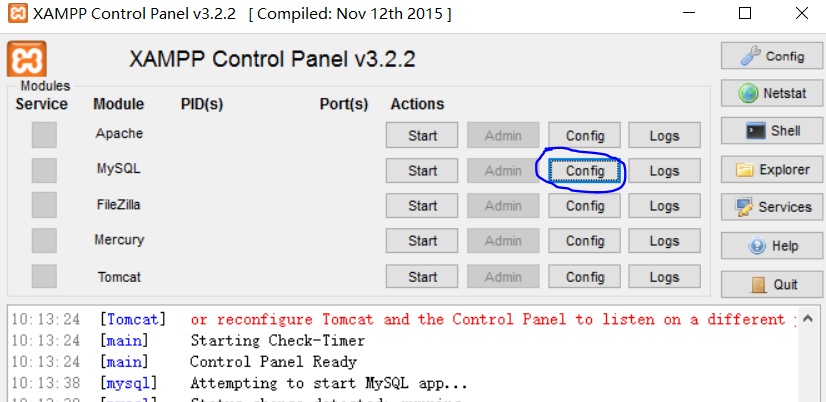
如果是查询的结果中,字段名中文的是乱码，则更新MySql.Data.dll,最好是6.4版本以上的，如果App. myDataAccess出错，则更新Mine.Data中的MySql.Data.dll。

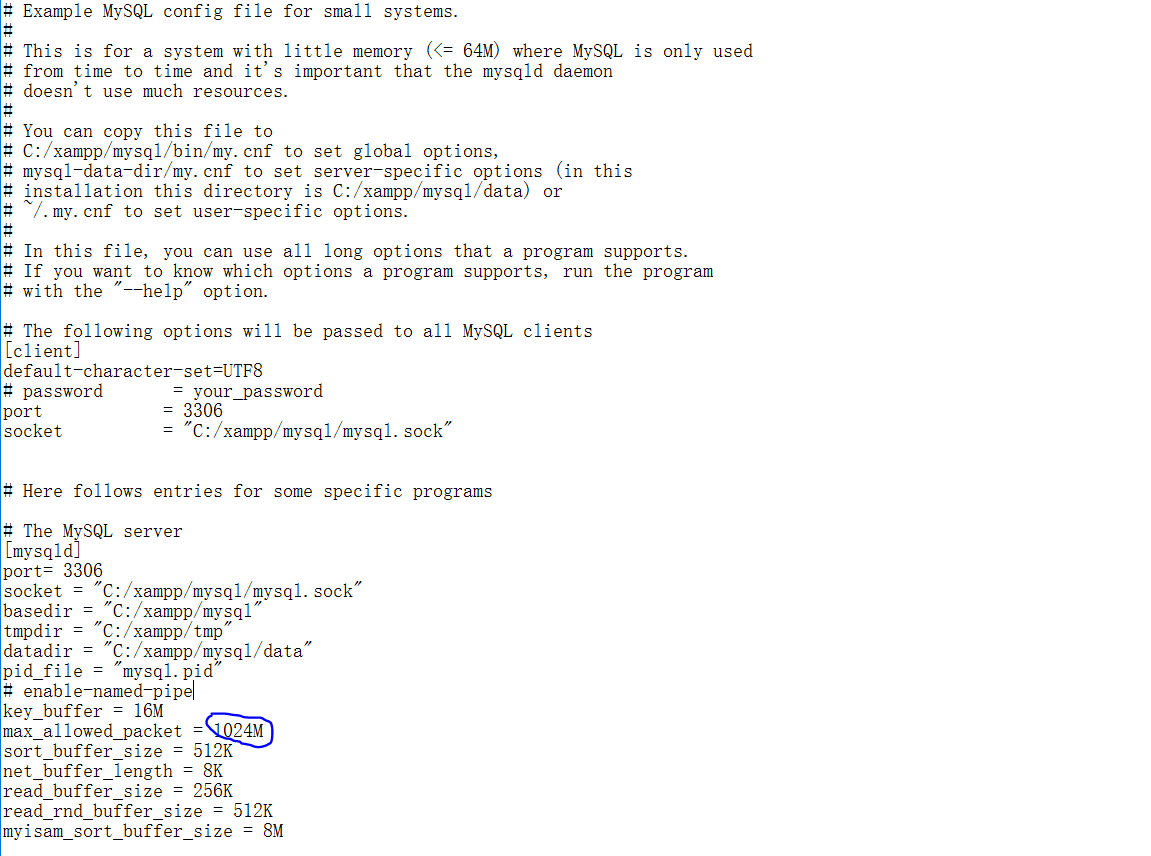
如果无法操作中文记录，显示为乱码，则连接字符串需要修改为：Database=kjgb\_data;Data Source=127.0.0.1;Port=3356;User Id=root;Password=kjgb2017;charset=utf8;

也就是加上charset=utf8;，同时Web.config中的<system.web> </system.web>中增加<globalization requestEncoding="utf-8" responseEncoding="utf-8"/>。

导入SQL文件提示” [Err] 2013 - Lost connection to MySQL server during query”的问题

配置MySQL允许的最大数据包大小。





windows下区分大小写的问题

Windows下mysql默认是不区分大小写的，但是在linux下是区分的。

my.ini 里面的mysqld部分

加入 lower\_case\_table\_names=2

[mysqld]  
lower\_case\_table\_names=2  
set-variable=lower\_case\_table\_names=0  
port= 3306

注: 1 表示不区分大小写 2表示区分大小写

然后重启mysqld服务器 ，F6。

mysql> show variables like '%low%';

+------------------------+---------+

| Variable\_name          | Value   |

+------------------------+---------+

| log\_slow\_queries       | OFF     |

| low\_priority\_updates   | OFF     |

| lower\_case\_file\_system | ON      |

| lower\_case\_table\_names | 2       |

| max\_allowed\_packet     | 1047552 |

| slow\_launch\_time       | 2       |

+------------------------+---------+

6 rows in set (0.06 sec)

mysql探究之null与not null

首先，我们要搞清楚“空值” 和 “NULL” 的概念：1、空值是不占用空间的；2、mysql中的NULL其实是占用空间的。

CREATE TABLE  `test` (

 `col1` VARCHAR( 10 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL ,

 `col2` VARCHAR( 10 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL

) ENGINE = MYISAM ;

插入数据：

INSERT INTO `test` VALUES (null,1);

mysql发生错误：

#1048 - Column 'col1' cannot be null

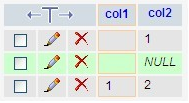
再来一条:

INSERT INTO `test` VALUES ('',1);

成功插入。可见，NOT NULL 的字段是不能插入“NULL”的，只能插入“空值”。

上面我们已经说过了，NULL 其实并不是空值，而是要占用空间，所以mysql在进行比较的时候，NULL 会参与字段比较，所以对效率有一部分影响。而且B树索引时不会存储NULL值的，所以如果索引的字段可以为NULL，索引的效率会下降很多。

SELECT \* FROM `test` WHERE col1 IS NOT NULL

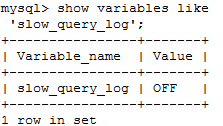


SELECT \* FROM `test` WHERE col1 <> ''



慢查询

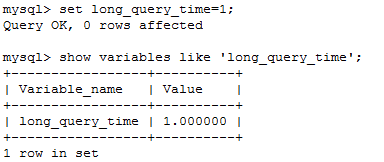
F6,mysql> show variables like 'slow\_query\_log';查看慢查询日志是否启用。



OFF，说明没有启用。



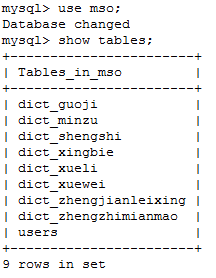
设置：将没有使用索引的查询记录日志。



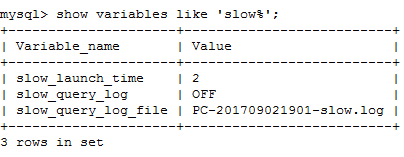
设置：查询时间超过1秒的查询记入慢查询日志。



设置：开启慢查询日志。



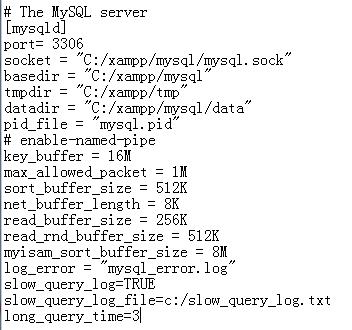
打开mso数据库，查询改数据库中的表。



查询慢查询日志记录的位置。

记录在：C:\xampp\mysql\data\ PC-201709021901-slow.log中。

命令行的方式暂时没有改过来，我是通过修改my.ini的方式修改的：



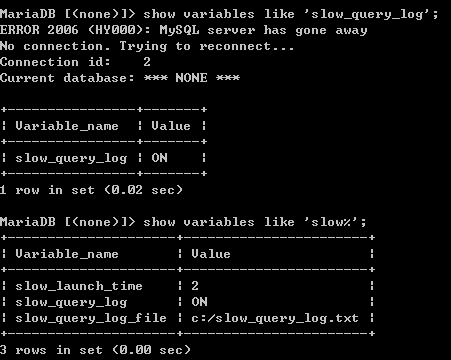
[mysqld]下加上最后三句：

slow\_query\_log=TRUE

slow\_query\_log\_file=c:/slow\_query\_log.txt

long\_query\_time=3

重启mysql后生效：



select \* from users limit 10;查询前10条，用时0.11秒，没有记录到日志中，select \* from users;用时10.026569秒，记录到日志中。

慢查询日志一般可以通过慢查询分析工具来查看，比如mysqldumpslow和pt-query-digest。mysqldumpslow和pt-query-digest都是perl脚本文件，要在perl环境下(windows下安装ActivePerl\_5.16.2.3010812913.msi)才能运行。mysql完整版内置了mysqldumpslow工具，pt-query-digest要去官网下。

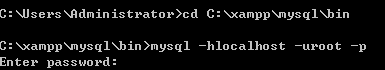
如：perl的安装目录是C:\Perl，我们添加环境变量，系统变量里的Path，添加C:\Perl\bin即可。

一般执行如下：

perl mysqldumpslow的路径\mysqldumpslow.pl+mysqldumpslow命令 +slow.log的路径+输出的地方

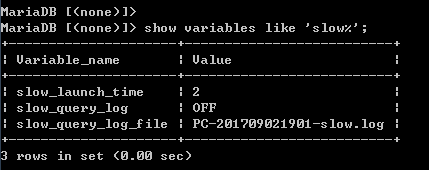


命令行里访问mysql库



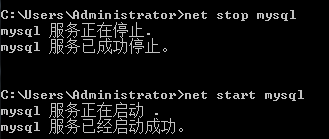
也可以直接带密码(下图中是kjgb2017)：





mysql重启

cmd中：



修改mysql密码

mysql> use mysql;

Database changed

mysql> update user set password=password("") where user="root";

Database changed

Rows matched: 4 Changed: 4 Warnings: 0

mysql> flush privileges;

Query OK, 0 rows affected

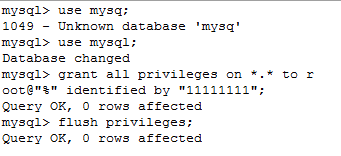
远程登录mysql

mysql -h localhost -P3307 -u root -p

也可以mysql -h localhost -P3307 -u root -p mytoubiaodata

mytoubiaodata是数据库名，3307是端口号。

默认无法远程登录mysql的解决方法



grant all privileges on \*.\* to root@"%" identified by "11111111";

flush privileges;

添加用户账号和权限

mysql -h localhost -P3307 -u root -p

也可以mysql -h localhost -P3307 -u root -p mytoubiaodata

mytoubiaodata是数据库名，3307是端口号。

会让输入登录密码，输入后回车即可。

create user cpdat;

use mysql;

CREATE USER 'x5'@'%' IDENTIFIED BY 'CpData';

CREATE USER 'x5'@'localhost' IDENTIFIED BY 'CpData';

//@"%" 表示对所有非本地主机授权，不包括localhost。（localhost地址设为127.0.0.1，如果设为真实的本地地址，不知道是否可以，没有验证。）

//对localhost授权：加上一句grant all privileges on testDB.\* to test@localhost identified by '1234';即可。

//所以以上要create两次。

//insert into mysql.user(Host,User,Password) values("%","cpdat",password("123456"));

//这里不要直接使用INSERT语句添加user记录，使用INSERT可能出现：ERROR 1364 (HY000): Field 'ssl\_cipher' doesn't have a default value错误。下句就可以创建用户：

GRANT USAGE ON \*.\* TO 'cpdat'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

GRANT SELECT ON cpdat.v\_project\_survey TO 'x5'@'%' IDENTIFIED BY 'CpData';

FLUSH PRIVILEGES;

上方\*.\*代表所有库的所有表的权限。

比如，以下是添加某视图权限给某用户的例子（已验证）：

navicat中按F6，FLUSH PRIVILEGES是让设置生效：

mysql> use mysql;

Database changed

mysql> CREATE USER 'x5'@'%' IDENTIFIED BY 'CpData';

Query OK, 0 rows affected (0.09 sec)

mysql> GRANT SELECT ON cpdat.v\_project\_survey TO 'x5'@'%' IDENTIFIED BY 'CpData';

Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;

Query OK, 0 rows affected (0.04 sec)

mysql> GRANT SELECT ON cpdat.v\_project\_survey TO 'x5'@'%' IDENTIFIED BY 'CpData';好像就是只给v\_project\_survey权限。

mysql自动备份

1. 批处理方式(可行)：

autobktoubiaodata.bat

E:

cd E:\投标分析决策系统数据库\mysql5.5\bin

mysqldump -uroot -pztsj mytoubiaodata > G:\12-投标分析决策系统\05-数据库\mytoubiaodata.sql

#pause

Exit

1. 创建一个计划任务

"开始菜单" -- > "控制面板" --> "管理工具" -- > "任务计划程序"

"Create A Basic Task" --> --- > "Daily" (注意设置后面的 时 分 秒) --> ---- "Start a Program" --> "Browser" (定位选择到你刚才创建的批处理mySql\_backup.bat) --> --- > "Finish"

强制使用索引

SELECT

tf.formId,

tf.formRemark,

tf.formTitle,

tf.isFinished,

tf.templateId,

tf.updateTime,

tf.latitude,

tf.longtitude,

tf.province,

tf.city,

tf.userId,

tm.MODEL\_NAME,

tp.PROJECT\_ID,

tp.PROJECT\_NAME,

tu.userName

FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp force INDEX(createrindex,PROSTRUCT\_ID\_index),

t\_user tu

mysql性能优化的步骤

(1)定位问题

a.开启慢查询日志，使用分析找出存在问题的查询。通常有以下几种情况：

-----查询次数多切每次查询占用时间长的SQL

通常为pt-query-digest(只能在linux下)分析的前几个查询。

-----IO大的SQL

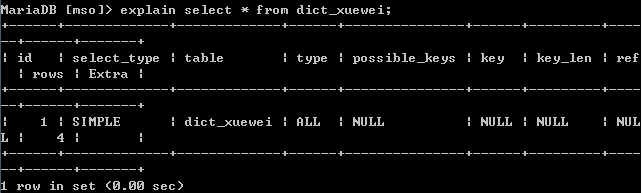
注意pt-query-digest分析中的Rows examine项。

-----未命中索引的SQL

注意pt-query-digest分析中的Rows examine和Rows Send的对比。

(2)explain从句：查询SQL的执行计划

数据库中的SQL都是先进行执行计划的分析，再进行具体的查询。



具体含义：

table:显示着一行的数据是关于哪张表的。

type:这是重要的列，显示连接使用了何种类型。从最好到最差的连接类型为：const(主键或唯一索引的查找)、eq\_reg(主键或唯一索引的范围查找)、ref(连接查询，一个表是基于某一个索引的查找)、range(索引的范围的查找)、index(索引的扫描)和ALL(表扫描)。

system > const > eq\_ref > ref > fulltext > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL

possible\_keys:显示可能应用在这张表上的索引。如果为空，没有可能的索引。

key:实际使用的索引。如果为NULL，就没有使用索引。

key\_len:使用的索引的长度。在不损失精确性的情况下，长度越短越好。

ref:显示索引的哪一列被使用了，如果可能的话，是一个常数。

rows:mysql认为必须检查的用来返回请求数据的行数。

use:扩展列，如下图中的Using where

Using filesort:看到这个的时候，查询就需要优化了。mysql需要进行额外的步骤来发现如何对返回的行排序。它根据连接类型以及存储排序键值和匹配条件的全部行的行指针来排序全部行。

Using temporary看到这个的时候，查询需要优化了。这里，mysql需要创建一个临时表来存储结果，这通常发生在对不同的列集进行order by上，而不是在group by上。

system 表只有一行：system表。这是const连接类型的特殊情况

const:表中的一个记录的最大值能够匹配这个查询（索引可以是主键或惟一索引）。因为只有一行，这个值实际就是常数，因为mysql先读这个值然后把它当做常数来对待

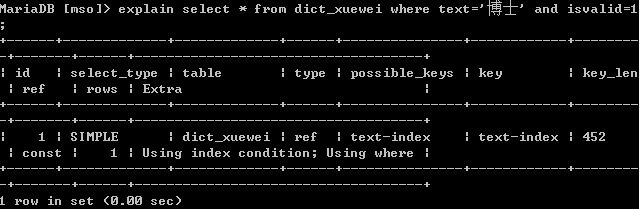
eq\_ref:在连接中，mysql在查询时，从前面的表中，对每一个记录的联合都从表中读取一个记录，它在查询使用了索引为主键或惟一键的全部时使用

ref:这个连接类型只有在查询使用了不是惟一或主键的键或者是这些类型的部分（比如，利用最左边前缀）时发生。对于之前的表的每一个行联合，全部记录都将从表中读出。这个类型严重依赖于根据索引匹配的记录多少—越少越好

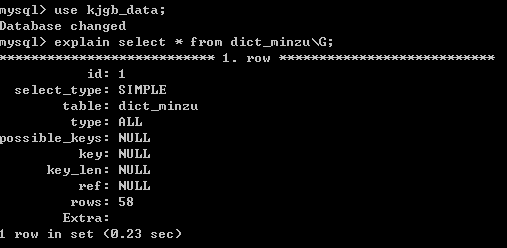
range:这个连接类型使用索引返回一个范围中的行，比如使用>或<查找东西时发生的情况

index: 这个连接类型对前面的表中的每一个记录联合进行完全扫描（比all更好，因为索引一般小于表数据）

all:这个连接类型对于前面的每一个记录联合进行完全扫描，这一般比较糟糕，应该尽量避免。



如果想结果整齐一些，竖排，可以加入\G。



不过在Navicat中的F6里无法这么做。

数据库层面的优化：

1. 选择合适的数据类型。

存下数据的最小的数据类型；

死锁

F6

use mysql;

show OPEN TABLES where In\_use > 0;

show processlist;

找到正在Query的Id。

kill Id; 如Id是420821, KILL 420821;

SQL查询需要注意的

1. 尽量避免在列上做运算，这样会导致索引失败；
2. 使用join是应该用小结果集驱动大结果集，同时把复杂的join查询拆分成多个query。不然join的越多表，就会导致越多的锁定和堵塞。
3. 注意like模糊查询的使用，避免使用%%，例如select \* from a where name like '%de%'; 代替语句：select \* from a where name >= 'de' and name < 'df';
4. 仅列出需要查询的字段，不要使用select \* from ...，节省内存；
5. 使用批量插入语句，节省交互；

insert into a (id ,name)

values(2,'a'),

(3,'s');

1. limit基数比较大时，使用between ... and ...
2. 避免使用null ，这就需要在建表时，尽量设置为not null，提升查询性能；
3. 不要使用count(id),而应该是count(\*)；
4. 不要做无谓的排序，尽可能在索引中完成排序；

Text类型的字段（对性能影响）

Text字段由于保存的内容多，因此在数据量比较大的情况下，会影响查询的效率。字段无数据情况下则查询较快。因此text字段建议单独一张表。

在查询时，如果查询的字段和where字段（含group by）和索引一样，则包含text字段的表查询较快，这是因为查询项不包含text字段，如果查询的字段和where字段（含group by）中存在非索引字段，则mysql会在查询索引结束后，将符合条件的数据行载入内存，这时text字段过大会造成IO的效率下降，从而导致查询效率的降低。

## mysql性能优化- profile

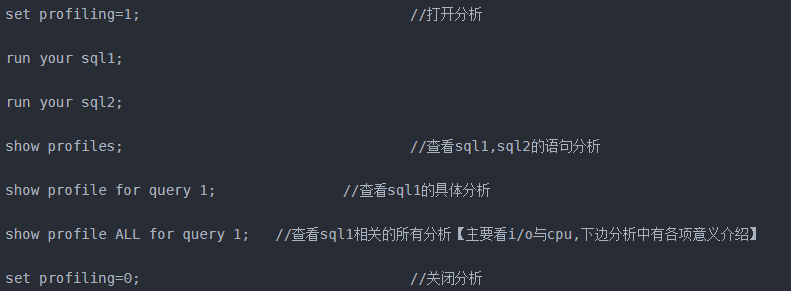
当我们要对某一条sql的性能进行分析时，可以使用它。

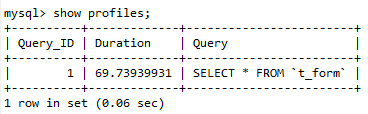
Profiling是从 mysql5.0.3版本以后才开放的。启动profile之后，所有查询包括错误的语句都会记录在内。关闭会话或者set profiling=0 就关闭了。

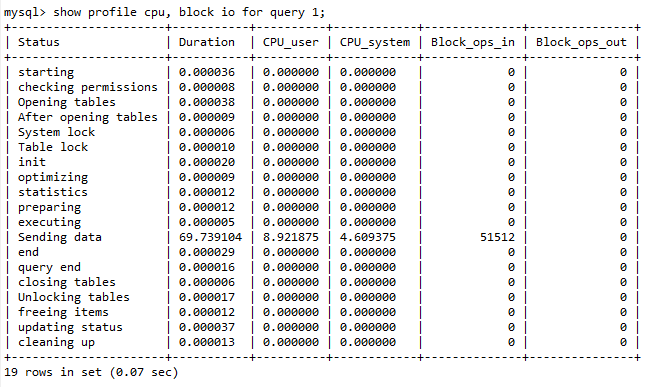
此工具可用来查询SQL执行状态，System lock和Table lock 花多少时间等等，对定位一条语句的I/O消耗和CPU消耗 非常重要。(SQL语句执行所消耗的最大两部分资源就是IO和CPU)。

在mysql5.7之后，profile信息将逐渐被废弃，mysql推荐使用performance schema

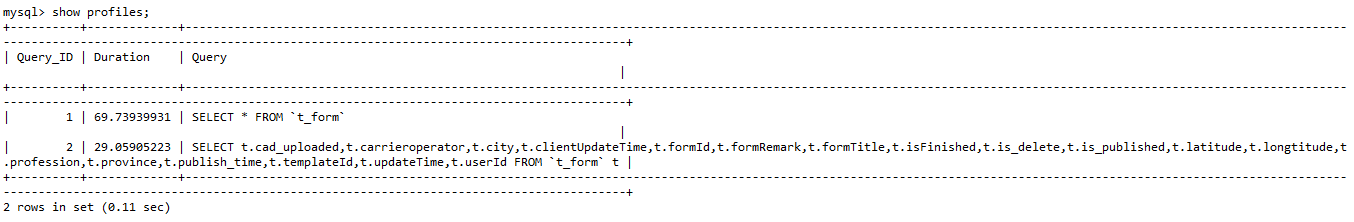
简易流程大概如下：





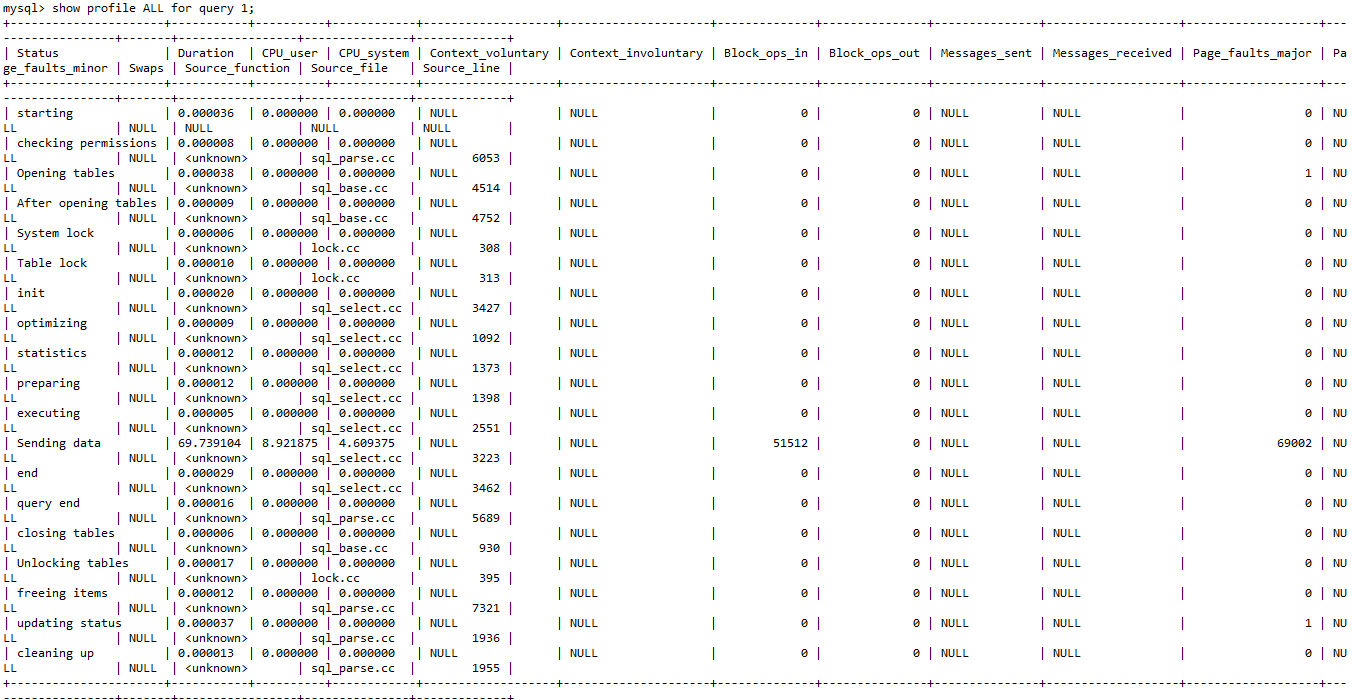


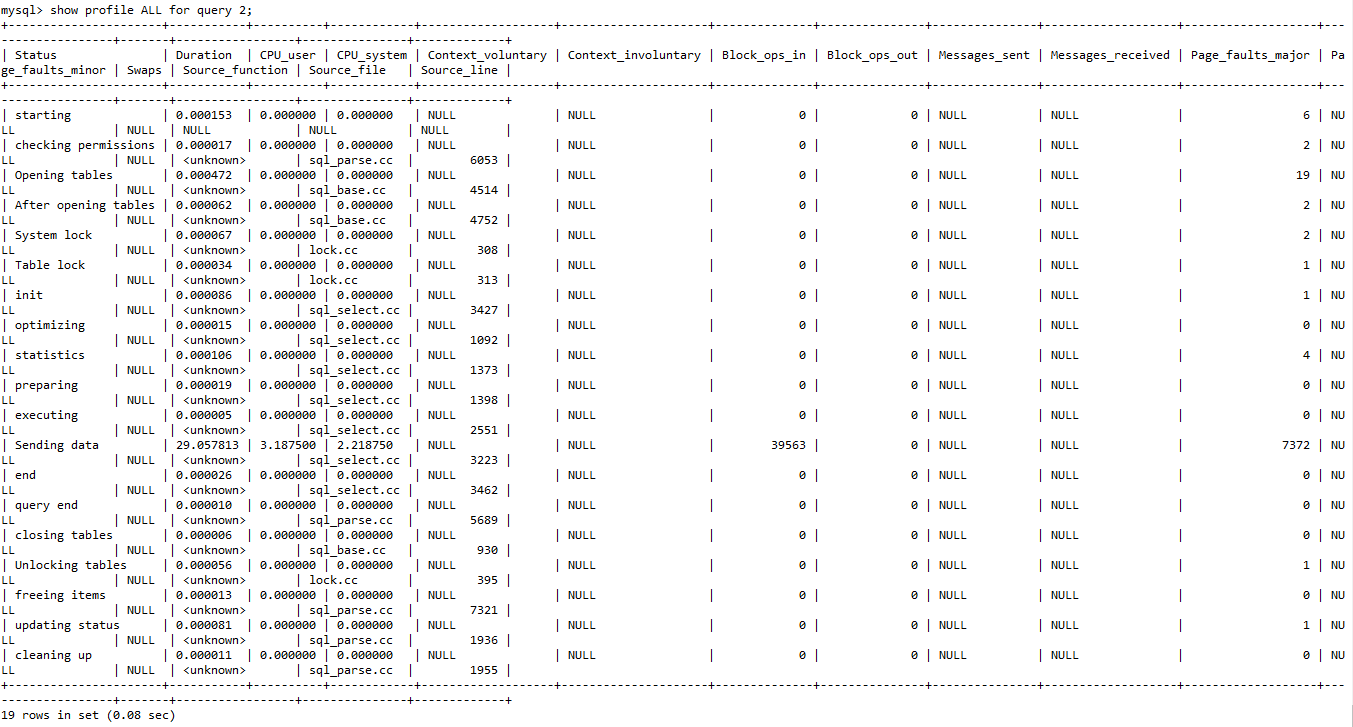
通过这个我们可以发现，主要集中在Sending data上。



可以看见，在缺少了body字段的条件下，效率提升了不少。

命令“show profile for query”的结果中有Sending data，该状态表示MySQL线程开始访问数据行并把结果返回给客户端，而不仅仅是返回结果给客户端。由于在Sending data状态下，MySQL线程往往需要做大量的磁盘读取操作，所以经常是整个查询中耗时最长的状态。





外键

cascade方式

在父表上update/delete记录时，同步update/delete掉子表的匹配记录。

set null方式

在父表上update/delete记录时，将子表上匹配记录的列设为null

要注意子表的外键列不能为not null。

No action方式

如果子表中有匹配的记录,则不允许对父表对应候选键进行update/delete操作。

Restrict方式

同no action, 都是立即检查外键约束。

Set default方式

父表有变更时,子表将外键列设置成一个默认的值 但Innodb不能识别。

常用查询语句

时间戳

10位：select FROM\_UNIXTIME(1156219870);

13位：select FROM\_UNIXTIME(1525231197135/1000);

差别是13位是用毫秒表示的，所以要/1000。

日期转时间戳：Select UNIX\_TIMESTAMP('2006-11-04 12:23:00');

一对多组合查询

select   c.name,  (select   group\_concat(name) from student   s   where  s.class\_id =c.id ) from  class  c

输出结果:

      三年二班           小明

     三年一班            小红,小兰

LIMIT

dict\_minzu中id从第20行(从0开始)开始往后2条记录。

SELECT \* FROM `dict\_minzu` ORDER BY id LIMIT 20,2;

SELECT \* FROM `dict\_minzu` ORDER BY id LIMIT 0,2;

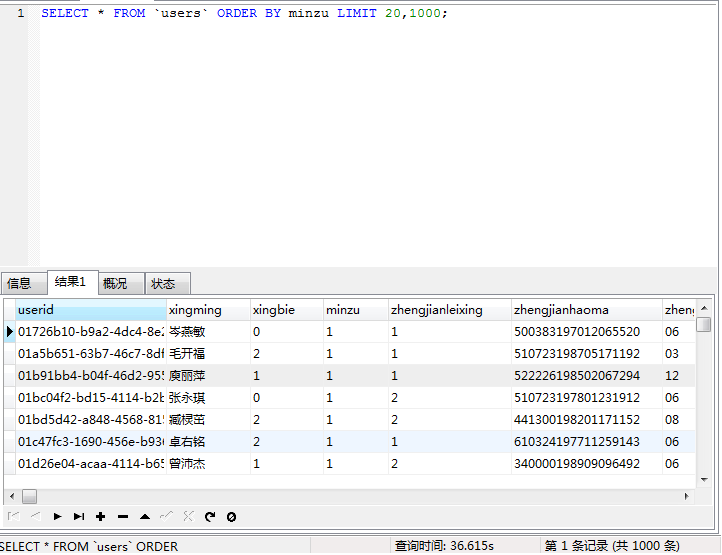
优化：

1. 使用有索引的列或主键进行order by操作。

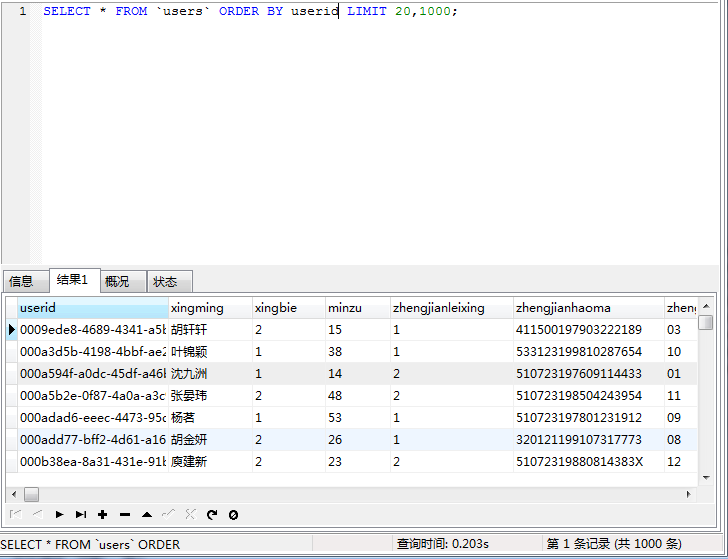
SELECT \* FROM `dict\_minzu` ORDER BY id LIMIT 20,2;当id是索引时，只会搜索22行，而不用索引，则会全表搜索排序。

不过当LIMIT 20000,2时，需要搜索20002条。

不使用索引时：



使用索引：



1. 记录上次查询的id

SELECT \* FROM `dict\_minzu` where id>19999 ORDER BY id LIMIT 1,2; 记录上次查询的id，但是对id是有要求的，如果id不连续，则可以添加一个自增的index列，对该列添加索引。

需要注意的问题：

SELECT

tf.formId,

tf.formRemark,

tf.formTitle,

tf.isFinished,

tf.templateId,

tf.updateTime,

tf.latitude,

tf.longtitude,

tf.province,

tf.city,

tf.userId,

tm.MODEL\_NAME,

tp.PROJECT\_ID,

tp.PROJECT\_NAME,

tu.userName

FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp force INDEX(createrindex,PROSTRUCT\_ID\_index),

t\_user tu

WHERE tf.is\_delete = 0 limit 0,30 --and tm.MODEL\_ID = tf.templateId and tp.PROJECT\_ID = tm.BELONG\_PROJECT and tf.userId = tu.userId and (tp.CREATER\_ID = 'liukai@tieta.com' OR tp.PROSTRUCT\_ID like '0000.0003.0015%') AND (tf.formTitle like '%河西%' or tp.PROJECT\_NAME like '%河西%') limit 0,30

如果将“河西”改成“陕西”，则查询结果很快出来，这个是因为陕西的重复项可能很多，很快就查询超过30条，则结果很快出来，但是河西的重复项较少，比如一共只有10条记录，数据库就会一直查询完。

MySQL中怎么对varchar类型排序问题

在mysql默认order by 只对数字与日期类型可以排序，但对于varchar字符型类型排序好像没有用了，下面我来给各位同学介绍varchar类型排序问题如何解决。

a.如果是电话，只要isdcode + 0，将它转换成int类型就可以了。

SELECT \* FROM gb\_country\_isdcode ORDER BY (isdcode+0) asc;

b.如果不是电话而是汉字怎么办，**汉字排序我们只要进行简单转换即可排序了**

在mysql中使用order by对存储了中文信息的字段，默认出来的结果并不是按汉字拼音的顺序来排序，要想按汉字的拼音来排序，需要把[数据库](http://www.111cn.net/list-55/)的字符集设置为UTF8，然后在order by 时候强制把该字段信息转换成GBK，这样出来的结果就是按拼音顺序排序的。例如：

SELECT \* FROM table\_name ORDER BY CONVERT(column\_name USING gbk);

select \* from 投标单位 where 是否有效='100' order by CONVERT(单位名称 USING gbk),排序号;

结论是：查询的时候，通过convert函数，把查询出来的数据使用的字符集gb2312编码就可以了，然后使用convert之后的中文排序。但是如果真的去把表中字段的字符集改成gb2312，又会涉及到很多编码的问题，页面传值啊，从[数据库](http://www.111cn.net/database/database.html)中存取啊，很麻烦。只要在查询的时候，指定一下字符集，并不是真的把物理字段改成gb2312，很简单。

查询结果前加序号

SELECT @rownum:=@rownum+1 AS 序号, t\_user.\* FROM t\_user ,(SELECT @rownum:=0) r

查询数据表行数

SELECT

DISTINCT substring(TB.TABLE\_NAME, 6) as 表名,TB.table\_rows AS 数量

FROM

INFORMATION\_SCHEMA.TABLES TB

WHERE

TB.TABLE\_SCHEMA = 'smart\_bid'

视图tables和statistics里面的统计信息缓存起来，所以数据未必准。

SELECT CONCAT( 'SELECT ', TABLE\_NAME, ', COUNT(\*) FROM ', TABLE\_SCHEMA, '.', TABLE\_NAME, ' UNION ALL' ) EXEC\_SQL

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_SCHEMA = 'smart\_bid'

然后将查询结果直接复制粘贴到记事本里，然后作为SQL查询。

# 三、SQLLite

# 四、SQLServer

# 五、Oracle

1、访问

命令行方式

sqlplus /nolog

conn [BIDUSER/CpData@172.16.0.72/orcl](mailto:BIDUSER/CpData@172.16.0.72/orcl)



查询语句记得加分号，否则sqlplus会认为语句没有结束，回车是换行，当前进入第二行，显示数字2。

Navicat

Instant Client 是Oracle 公司推出的精简版客户端。

32位下载地址：<http://www.oracle.com/technetwork/topics/winsoft-085727.html>

注意：不管你的Navicat是32位的还是64位的，你的Instant Client都要下载32位的，因为Navicat只支持32位的客户端。另外要下载 Basic 版本 (40M)，而不是 Basic Lite 版(11兆)，因为 Basic Lite 版不支持中文，连接的时候会告诉你服务器端的编码不被支持。

还有一点要注意，Oracle9i或以上版本的，要安装Install Client11或以下；Oracle8或8i服务器，需要安装Install Client10或以下。这个问题不大，因为我们现在的Oracle都是10或11了，注意一下就好。

Navicat Premium默认自带了instantclient\_10\_2，但是其是base lite版本的（Basic Lite： Basic 的精简版本，其中仅带有英文错误消息和 Unicode、ASCII 以及西欧字符集支持），不支持中文字符集，而本文中的服务器端oracle恰好是中文字符集，instantclient\_10\_2不适合，因此需要下载base版本。

2、常见错误

1.ora-12514:

listener.ora:

SID\_LIST\_LISTENER =

(SID\_LIST =

(SID\_DESC =

(SID\_NAME = PLSExtProc)

(ORACLE\_HOME = C:\app\Administrator\product\11.1.0\db\_1)

(PROGRAM = extproc)

)

(SID\_DESC =

(GLOBAL\_DBNAME = ORCL)

(ORACLE\_HOME = C:\app\Administrator\product\11.1.0\db\_1)

(SID\_NAME = ORCL)

)

)

LISTENER =

(DESCRIPTION\_LIST =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))

(ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))

)

)

3、查询语句和函数：

1.查询oracle某用户下的表名和空间名：

select TABLE\_NAME,TABLESPACE\_NAME from dba\_tables where owner='TESTDS'

2.查看数据库当前所有会话

SELECT \* FROM V$SESSION WHERE USERNAME IS NOT NULL ORDER BY LOGON\_TIME , SID;

3.数据库最大连接数

SELECT SESSIONS\_MAX,SESSIONS\_WARNING,SESSIONS\_CURRENT,SESSIONS\_HIGHWATER FROM v$license;

4. group by分组统计：

select t.一级学科id,count(t.二级学科id) from 二级学科 t group by t.一级学科id order by t.一级学科id

select xk.一级学科id,xk.一级学科名称,count(t.二级学科id) from 二级学科 t inner join 一级学科 xk on xk.一级学科id=t.一级学科id group by xk.一级学科id,xk.一级学科名称 order by xk.一级学科id

5.根据出生日期查询年龄

(to\_char(sysdate,'YYYY')-to\_char(grxx.出生日期, 'YYYY')) as 年龄,

或

trunc(months\_between(sysdate, xx.出生年月日) / 12) as 年龄

6.字段查重

select 身份证号,count(\*) from A group by 身份证号 having count(\*)>1

7.姓名处理(两个字的姓名中间加空格)

(case when length(jjsq.姓名)=2 then (substr(jjsq.姓名,1,1) || '　' || substr(jjsq.姓名,2,1)) else jjsq.姓名 end) as姓名

8.查看两小时内，IIS连接数据导致的非活动进程

SELECT SID, SERIAL#,MODULE, STATUS

FROM V$SESSION S

WHERE S.USERNAME IS NOT NULL

AND UPPER(S.PROGRAM) IN ('TOAD.EXE', 'W3WP.EXE')

AND S.LAST\_CALL\_ET >= 60\*60\*2

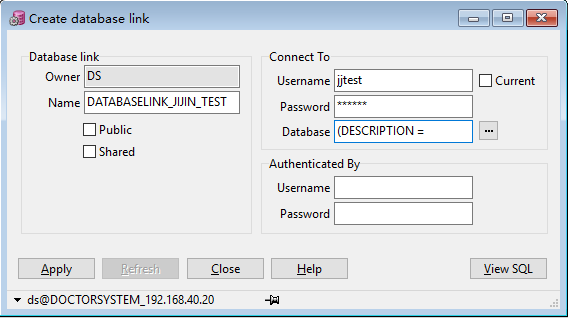
AND S.STATUS = 'INACTIVE'

ORDER BY SID DESC;

4、PL/SQL：

1.添加Database links

右击选择“新增”，如下图：



点击“Apply”即可，Database一栏中填写的内容如下：

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.13)(PORT = 1521))

)

(CONNECT\_DATA =

(SERVICE\_NAME = orcl)

)

)

尽量不要用服务名，服务名虽然可以，但是换到另外一台机器可能不行(因为服务名必须在要在数据库安装的服务器上存在该服务名才行，且指向必须正确)。

如果上述文字无法复制到文本框中，点击DataBase旁边的按钮，就可以在弹出的界面复制该文本了。

5、日志和维护：

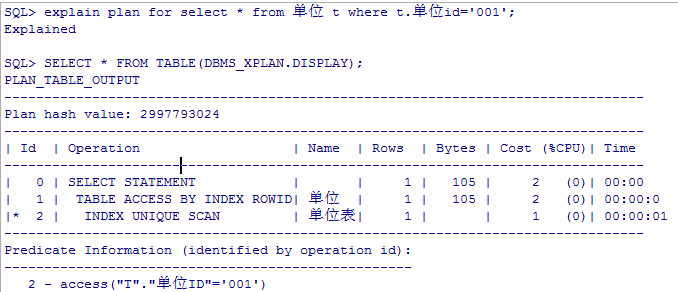
1.归档

想要完整日志，必须要归档模式，且硬盘得大。

6、性能调优：

1. explain plan for

和mysql的explain类似，oracle中也有查询计划，使用如下：



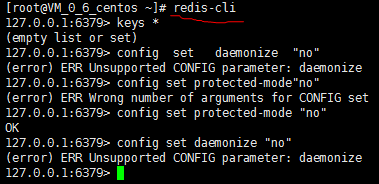
7、Navicat：

无需安装客户端，直接下载对应版本的oci，只要大版本和window64位或32位一致，在Navicat-工具-选项中对oci进行一下关联就可以。

<https://blog.csdn.net/gxp1182893781/article/details/79815573>

# 六、Redis

1、进入redis



redis-cli

keys \* //查询所有key的值。

quit //退出redis

## 2、远程连接redis

redis-cli -h 132.232.6.96  -p  6379

shift+右键：在此处打开命令窗口。

如果无法连接，

1.注释掉redis.windows-service.conf 中的bind 127.0.0.1这一行(在前面加#)

2.同文件中将protected-mode yes  改成  protected-mode no

3.保存并重启redis服务

在redis的配置文件redis.conf中，vi redis.conf，找到bind localhost注释掉。

配置文件一般在/etc中：

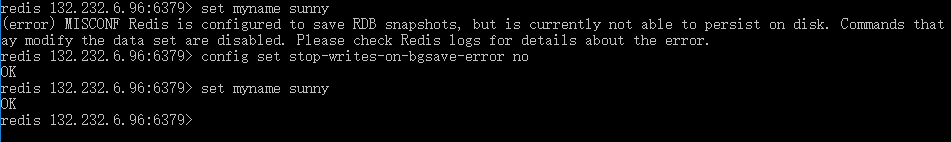




注释掉本机,局域网内的所有计算机都能访问。

bind localhost   只能本机访问,局域网内计算机不能访问。

bind  局域网IP    只能局域网内IP的机器访问, 本地localhost都无法访问。



提示错误：

(error) MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk. Commands that may modify the data set are disabled. Please check Redis logs for details about the error.

Redis被配置为保存数据库快照，但它目前不能持久化到硬盘。用来修改集合数据的命令不能用。请查看Redis日志的详细错误信息。

解决方法：

运行config set stop-writes-on-bgsave-error no　命令后，关闭配置项stop-writes-on-bgsave-error解决该问题。

见上图。

## 3、设置redis密码

Redis默认配置是不需要密码认证的，也就是说只要连接的Redis服务器的host和port正确，就可以连接使用。这在安全性上会有一定的问题，所以需要启用Redis的认证密码，增加Redis服务器的安全性。

(1)修改配置文件

Redis的配置文件默认在/etc/redis.conf，找到如下行：

#requirepass foobared

去掉前面的注释，并修改为所需要的密码：

requirepass myPassword （其中myPassword就是要设置的密码）

(2)重启redis服务

如果Redis已经配置为service服务，可以通过以下方式重启：

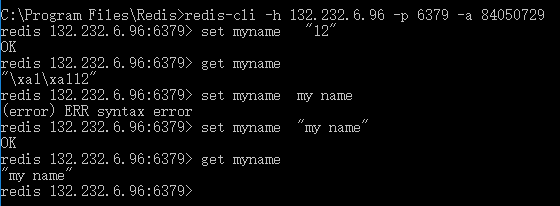
service redis restart

如果Redis没有配置为service服务，可以通过以下方式重启：

/usr/local/bin/redis-cli shutdown

/usr/local/bin/redis-server /etc/redis.conf

(3)远程连接redis服务



84050729就是redis的密码。

4、Redis客户端的GUI工具

https://blog.csdn.net/it\_0101/article/details/77745109

99、redis和memcached比较

详见《两大数据库缓存系统代码级实现对比》。

# 九、大数据

1、solr

solr 是一个全文搜索工具，它是由Lucene发展而来。那么，solr 和 Lucene 之间是什么样的关系呢？简单地说，如果把solr比喻成一辆汽车，那么Lucene就是一台汽车引擎。绝大多数情况下我们完全可以使用solr 来解决开发中遇到的问题。

安装

解压solr-6.3.0.zip 文件到 E:\blog\test1\solr-6.3.0。用管理员身份进入命令行，用cd命令进入 E:\blog\test1\solr-6.3.0\bin 文件夹下，输入命令：

solr -e dih

就可以启动 solr 服务器了。打开浏览器，输入网址：localhost:8983/solr/ 。你就可以进入solr的WEB控制台。   
如果想要停止solr服务器，输入命令：

solr stop -all



# 十、效率

1、联合查询

多表联合查询可用如下语法，效率快很多：

SELECT

tu.userName RANK\_NAME,

rank.taskNum TASK\_NUM

FROM

(

SELECT

tf.userId,

COUNT(tf.userId) taskNum

FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp

WHERE tm.MODEL\_ID = tf.templateId and tp.PROJECT\_ID = tm.BELONG\_PROJECT and (updateTime > 1514736000000 and updateTime < 1522339200000 and 1=1) and tf.is\_delete = 0

GROUP BY

tf.userId

) rank,

t\_user tu

WHERE

rank.userId = tu.userId

ORDER BY

rank.taskNum DESC

优化得比较明显的：

SELECT d.\* from ((SELECT

tf.formId,

tf.formRemark,

tf.formTitle,

tf.isFinished,

tf.templateId,

tf.updateTime,

tf.latitude,

tf.longtitude,

tf.province,

tf.city,

tf.userId,

tm.MODEL\_NAME,

tp.PROJECT\_ID,

tp.PROJECT\_NAME,

tu.userName

FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp,

t\_user tu

WHERE tf.is\_delete = 0 and tm.MODEL\_ID = tf.templateId and tp.PROJECT\_ID = tm.BELONG\_PROJECT and tf.userId = tu.userId and (tp.PROSTRUCT\_ID like '0000.0003.0009\_32%' or tp.PROSTRUCT\_ID like '0000.0003.0009%' or tp.CREATER\_ID='889151'))

union ALL

(

SELECT

tf.formId,

tf.formRemark,

tf.formTitle,

tf.isFinished,

tf.templateId,

tf.updateTime,

tf.latitude,

tf.longtitude,

tf.province,

tf.city,

tf.userId,

tm.MODEL\_NAME,

tp.PROJECT\_ID,

tp.PROJECT\_NAME,

tu.userName

FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp,

t\_user tu

WHERE tf.is\_delete = 0 and tm.MODEL\_ID = tf.templateId and tp.PROJECT\_ID = tm.BELONG\_PROJECT and tf.userId = tu.userId and tf.userId = '889151'

) ) d where d.formTitle like '%南京%' or d.PROJECT\_NAME like '%南京%'

ORDER BY updateTime DESC

总结几点：

1.多表联合查询中，如果最后有多表的条件or连接，尽量拆成union，如上表中的tp.CREATER\_ID='889151'和tf.userId = '889151'，用or连在一起，索引就失效了，分开查询用union就可以。

2.union去掉重复项，union all不会。

3.左从模糊查询索引是有效的，如：like ‘\*\*\*\*\*%’，但是like ‘%\*\*\*\*\*%’，索引就失效了，查询很慢。

4. FROM

t\_form tf,

ttsurvay\_model tm,

t\_project tp,

t\_user tu

这种写法比left join的写法要快不少。