# Mysql

https://blog.csdn.net/qq\_30450439/article/details/80089892

mysqld服务器维护两种变量。全局变量影响服务器的全局操作。会话变量影响具体客户端连接相关操作。

服务器启动时，将所有全局变量初始化为默认值。可以在选项文件或命令行中指定的选项来更改这些默认值。服务器启动后，通过连接服务器并执行SET GLOBAL var\_name语句可以更改动态全局变量。要想更改全局变量，必须具有SUPER权限。

服务器还为每个客户端连接维护会话变量。连接时使用相应全局变量的当前值对客户端会话变量进行初始化。客户可以通过SET SESSION var\_name语句来更改动态会话变量。设置会话变量不需要特殊权限，但客户可以只更改自己的会话变量，而不更改其它客户的会话变量。

任何访问全局变量的客户端都可以看见对全局变量的更改。但是，它只影响在更改后连接的从该全局变量初始化相应会话变量的客户端。它不会影响已经连接上的客户端的会话变量(甚至是执行SET GLOBAL语句的客户端)。

bdb\_cache\_size

分配给BDB类型数据表的缓存索引和行排列的缓冲区大小，如果不使用DBD类型数据表，则应该在启动MySQL时加载 --skip-bdb 参数以避免内存浪费。

bdb\_log\_buffer\_size

分配给BDB类型数据表的缓存索引和行排列的缓冲区大小，如果不使用DBD类型数据表，则应该将该参数值设置为0，或者在启动MySQL时加载 --skip-bdb 参数以避免内存浪费。

# trim函数可以过滤指定的字符串：

完整格式：TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)

简化格式：TRIM([remstr FROM] str)

返回字符串 *str* ， 其中所有*remstr*前缀和/或后缀都已被删除。若分类符BOTH、LEADIN或TRAILING中没有一个是给定的,则假设为BOTH。*remstr*为可选项，在未指定情况下，可删除空格。

**[sql]** [view plain](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "view plain) [copy](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "copy)

1. mysql> **SELECT** TRIM('  bar   ');
3. -> 'bar'
5. mysql> **SELECT** TRIM(LEADING 'x' **FROM** 'xxxbarxxx');   --删除指定的首字符 x
7. -> 'barxxx'
9. mysql> **SELECT** TRIM(BOTH 'x' **FROM** 'xxxbarxxx');      --删除指定的首尾字符 x
11. -> 'bar'
13. mysql> **SELECT** TRIM(TRAILING 'xyz' **FROM** 'barxxyz');  --删除指定的尾字符 x
15. -> 'barx'

**mysql中的去除左空格函数**:

LTRIM(str);

**[sql]** [view plain](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "view plain) [copy](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "copy)

1. mysql> **SELECT** LTRIM('  barbar');
3. -> 'barbar'

**mysql中的去除右空格函数：**

RTRIM(str):

**[sql]** [view plain](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "view plain) [copy](https://blog.csdn.net/jiangnan2014/article/details/21948279" \o "copy)

1. mysql> **SELECT** RTRIM('barbar   ');
3. -> 'barbar'

# [MySQL CAST与CONVERT 函数的用法](https://www.cnblogs.com/chenqionghe/p/4675844.html)

MySQL 的CAST()和CONVERT()函数可用来获取一个类型的值，并产生另一个类型的值。两者具体的语法如下：

|  |  |
| --- | --- |
| 1  2 | CAST(value as type);  CONVERT(value, type); |

就是CAST(xxx AS 类型), CONVERT(xxx,类型)。

可以转换的类型是有限制的。这个类型可以是以下值其中的一个：

* 二进制，同带binary前缀的效果 : BINARY
* 字符型，可带参数 : CHAR()
* 日期 : DATE
* 时间: TIME
* 日期时间型 : DATETIME
* 浮点数 : DECIMAL
* 整数 : SIGNED
* 无符号整数 : UNSIGNED

下面举几个例子：

例一

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | mysql> SELECT CONVERT('23',SIGNED);  +----------------------+  | CONVERT('23',SIGNED) |  +----------------------+  |                   23 |  +----------------------+  1 row in set |

例二

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | mysql> SELECT CAST('125e342.83' AS signed);  +------------------------------+  | CAST('125e342.83' AS signed) |  +------------------------------+  |                          125 |  +------------------------------+  1 row in set |

例三

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | mysql> SELECT CAST('3.35' AS signed);  +------------------------+  | CAST('3.35' AS signed) |  +------------------------+  |                      3 |  +------------------------+  1 row in set |

像上面例子一样，将varchar 转为int 用 cast(a as signed)，其中a为varchar类型的字符串。

例4

在SQL Server中，下面的代码演示了datetime变量中，仅包含单纯的日期和单纯的时间时，日期存储的十六进制存储表示结果。

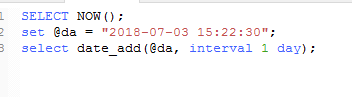
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | DECLARE @dt datetime    --单纯的日期  SET @dt='1900-1-2'  SELECT CAST(@dt as binary(8))  --结果: 0x0000000100000000    --单纯的时间  SET @dt='00:00:01'  SELECT CAST(@dt as binary(8))  --结果: 0x000000000000012C |

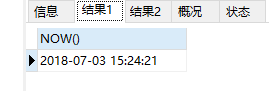
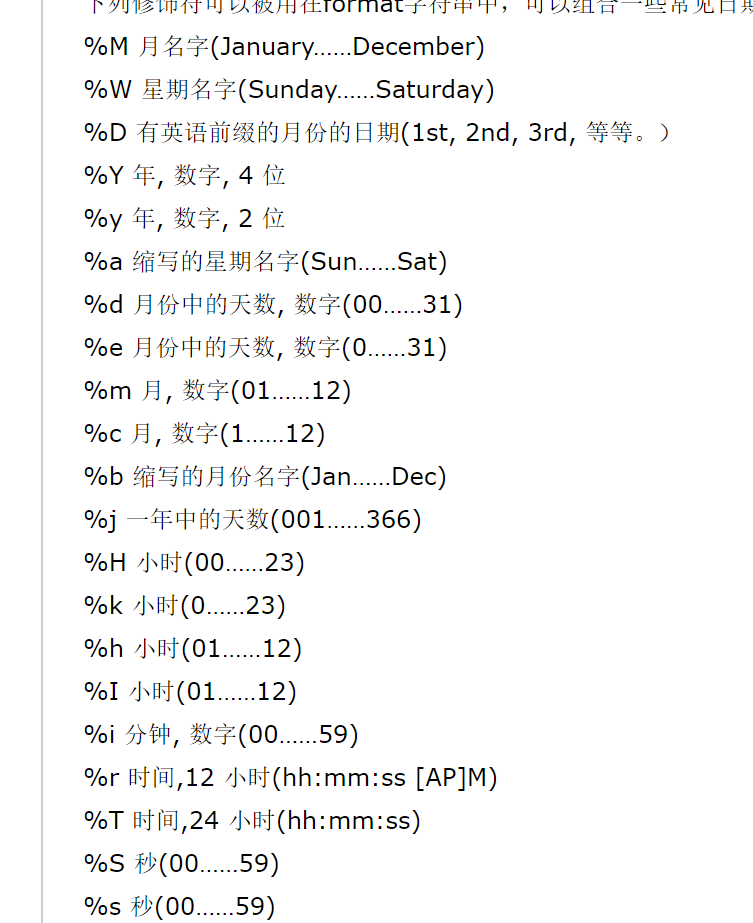
MySQL的类型转换和SQL Server一样，就是类型参数有点点不同：CAST(xxx AS 类型) ,CONTVER(xxx,类型)。

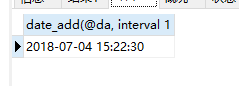
# 操作手册

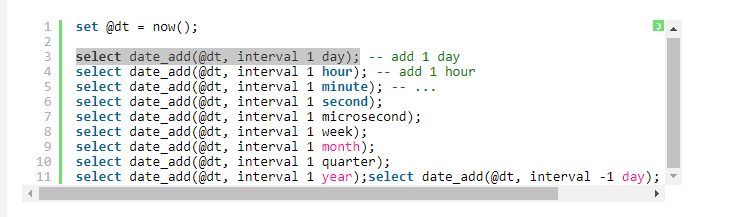
* <http://blog.sina.com.cn/s/blog_52d20fbf0100ofd5.html>
* <https://blog.csdn.net/a__yes/article/details/52795793>
* <https://dev.mysql.com/doc/refman/8.0/en/information-schema.html> (mysql官方文档)
* <http://blog.51cto.com/imysqldba/592068> （mysql实现数组）

# 日期滚动









更多：https://www.jb51.net/article/100897.htm

# LEFT

LFET（s，n）从左开始截取s字符串，n字符

类似于left这种关键字为名的方法，Navicat会提示使用`left`()。但实际运行会报错。

# 针对获取所有更新字段的sql

SELECT GROUP\_CONCAT(DISTINCT CONCAT("\"", COLUMN\_NAME,"=\",",COLUMN\_NAME)) from information\_schema.`COLUMNS` WHERE TABLE\_NAME = "dc\_vehicle\_maintenance\_taskinfo"

# 创建对象的sql

SELECT

CONCAT(

"private ",

CASE

WHEN DATA\_TYPE = "varchar" THEN

"String"

WHEN DATA\_TYPE = "bigint" THEN

"long"

ELSE

DATA\_TYPE

END,

" ",

COLUMN\_NAME,";"

)

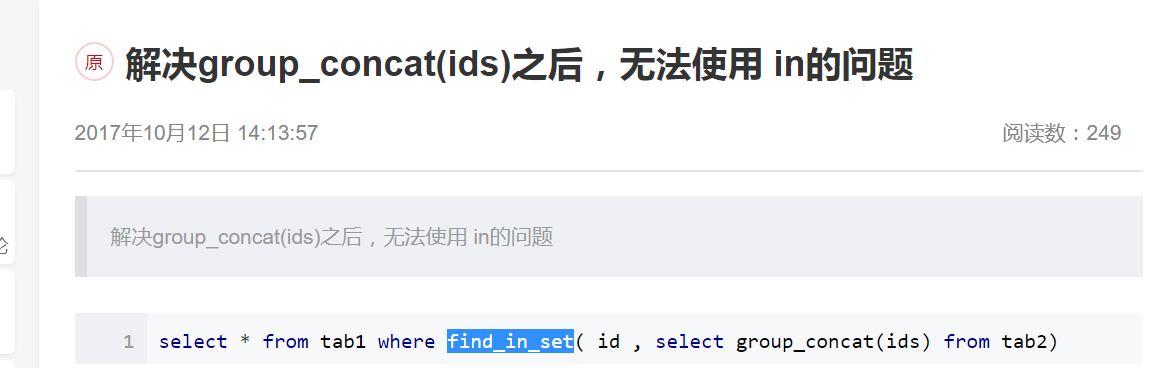
FROM

information\_schema.`COLUMNS`

WHERE

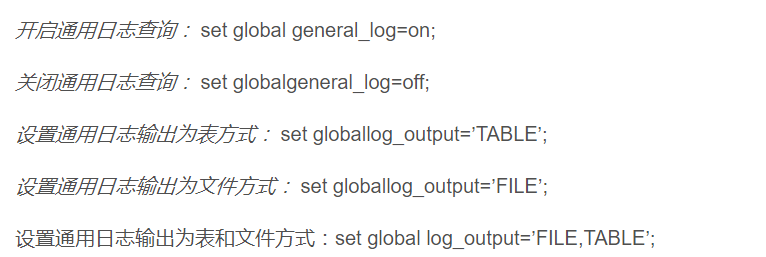
TABLE\_NAME = "iv4dm\_alarmfeedback"

# 解决group\_concat(ids)之后，无法使用 in的问题



# Mysql慢查询

https://blog.csdn.net/timchen525/article/details/75268151



# Mysql String函数

|  |  |
| --- | --- |
| 函数 | 功能 |
| CANCAT(S1,S2,…Sn) | 连接 S1,S2,…Sn 为一个字符串 |
| INSERT(str,x,y,instr) | 将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr |
| LOWER(str) | 将字符串 str 中所有字符变为小写 |
| UPPER(str) | 将字符串 str 中所有字符变为大写 |
| LEFT(str ,x) | 返回字符串 str 最左边的 x 个字符 |
| RIGHT(str,x) | 返回字符串 str 最右边的 x 个字符 |
| LPAD(str,n ,pad) | 用字符串 pad 对 str 最左边进行填充，直到长度为 n 个字符长度 |
| RPAD(str,n,pad) | 用字符串 pad 对 str 最右边进行填充，直到长度为 n 个字符长度 |
| LTRIM(str) | 去掉字符串 str 左侧的空格 |
| RTRIM(str) | 去掉字符串 str 行尾的空格 |
| REPEAT(str,x) | 返回 str 重复 x 次的结果 |
| REPLACE(str,a,b) | 用字符串 b 替换字符串 str 中所有出现的字符串 a |
| STRCMP(s1,s2) | 比较字符串 s1 和 s2（比较ASCII 码值） |
| TRIM(str) | 去掉字符串行尾和行头的空格 |
| SUBSTRING(str,x,y) | 返回从字符串 str x 位置起 y 个字符长度的字串 |
| SUBSTRING\_INDEX（str,key,index） | 以第index个key为切割 |

# Mysql 数值函数

5-2 MySQL 中的常用数值函数

|  |  |
| --- | --- |
| 函数 | 功能 |
| ABS(x) | 返回 x 的绝对值 |
| CEIL(x) | 返回大于 x 的最大整数值 |
| FLOOR(x) | 返回小于 x 的最大整数值 |
| MOD(x，y) | 返回 x/y 的模 |
| RAND() | 返回 0 到 1 内的随机值 |
| ROUND(x,y) | 返回参数 x 的四舍五入的有 y 位小数的值 |
| TRUNCATE(x,y) | 返回数字 x 截断为 y 位小数的结果 |

# Mysql 日期函数

|  |  |
| --- | --- |
| 函数 | 功能 |
| CURDATE() | 返回当前日期 |
| CURTIME() | 返回当前时间 |
| NOW() | 返回当前的日期和时间 |
| UNIX\_TIMESTAMP(date) | 返回日期 date 的 UNIX 时间戳 |
| FROM\_UNIXTIME | 返回 UNIX 时间戳的日期值 |
| WEEK(date) | 返回日期 date 为一年中的第几周 |
| YEAR(date) | 返回日期 date 的年份 |
| HOUR(time) | 返回 time 的小时值 |
| MINUTE(time) | 返回 time 的分钟值 |
| MONTHNAME(date) | 返回 date 的月份名 |
| DATE\_FORMAT(date,fmt) | 返回按字符串 fmt 格式化日期 date 值 |
| DATE\_ADD(date,INTERVAL expr type) | 返回一个日期或时间值加上一个时间间隔的时间值 |
| DATEDIFF(expr,expr2) | 返回起始时间 expr 和结束时间 expr2 之间的天数 |

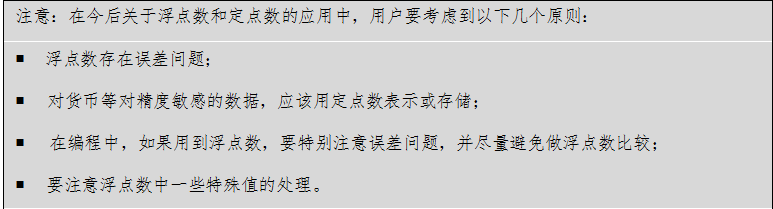
MySQL 中的日期时间格式

|  |  |
| --- | --- |
| 格式符 | 格式说明 |
| %S,%s | 两位数字形式的秒（00,01,...,59） |
| %i | 两位数字形式的分（00,01,...,59） |
| %H | 两位数字形式的小时，24 小时（00,01,...,23） |
| %h,%I | 两位数字形式的小时，12 小时（01,02,...,12） |
| %k | 数字形式的小时，24 小时（0,1,...,23） |
| %l | 数字形式的小时，12 小时（1,2,...,12） |
| %T | 24 小时的时间形式（hh:mm:ss） |
| %r | 12 小时的时间形式（hh:mm:ssAM 或 hh:mm:ssPM） |
| %p | AM 或 PM |
| %W | 一周中每一天的名称（Sunday,Monday,...,Saturday） |
| %a | 一周中每一天名称的缩写（Sun,Mon,...,Sat） |
| %d | 两位数字表示月中的天数（00,01,...,31） |
| %e | 数字形式表示月中的天数（1,2，...,31） |
| %D | 英文后缀表示月中的天数（1st,2nd,3rd,...） |
| %w | 以数字形式表示周中的天数（0=Sunday,1=Monday,...,6=Saturday） |
| %j | 以 3 位数字表示年中的天数（001,002,...,366） |
| %U | 周（0,1,52），其中 Sunday 为周中的第一天 |
| %u | 周（0,1,52），其中 Monday 为周中的第一天 |
| %M | 月名（January,February,...,December） |

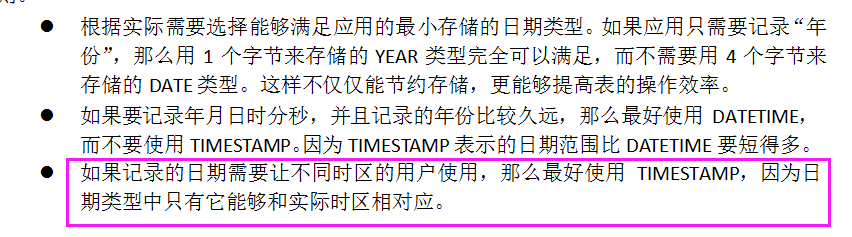
# **MySQL 流程函数**

|  |  |
| --- | --- |
| 函数 | 功能 |
| IF(value,t, f) | 如果 value 是真，返回 t；否则返回 f |
| IFNULL(value1,value2) | 如果 value1 不为空返回 value1，否则返回 value2 |
| CASE WHEN [value1]  THEN[result1]…ELSE[default]END | 如果 value1 是真，返回 result1，否则返回 default |
| CASE [expr] WHEN [value1]  THEN[result1]…ELSE[default]END | 如果 expr 等于 value1，返回 result1，否则返回 default |

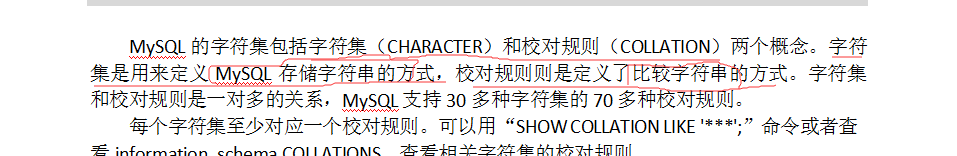
# **Mysql定点数，浮点数**

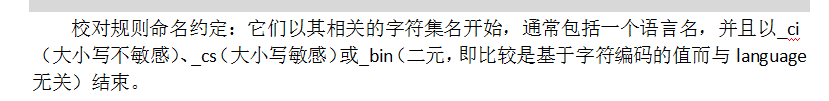


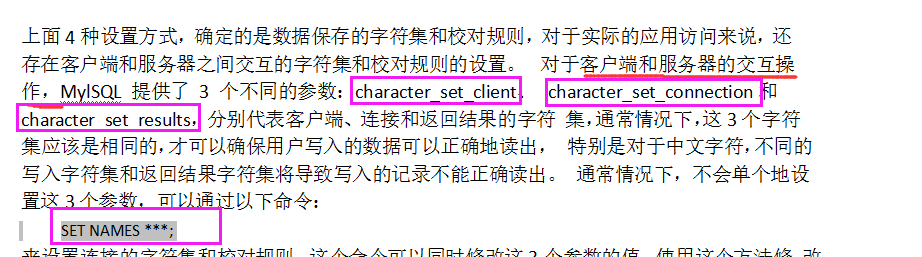
# Mysql 日期的选择



# 字符集选择







# 字符集的修改步骤

（1）导出表结构：

mysqldump -uroot -p --default-character-set=gbk -d databasename> createtab.sql

其中--default-character-set=gbk 表示设置以什么字符集连接，-d 表示只导出表结构， 不导出数据。

（2）手工修改 createtab.sql 中表结构定义中的字符集为新的字符集。

（3）确保记录不再更新，导出所有记录。

mysqldump -uroot -p --quick --no-create-info --extended-insert

--default-character-set=latin1 databasename> data.sql

 --quick：该选项用于转储大的表。它强制 mysqldump 从服务器一次一行地检索表中 的行而不是检索所有行，并在输出前将它缓存到内存中。

* + - --extended-insert：使用包括几个 VALUES 列表的多行 INSERT 语法。这样使转储文件 更小，重载文件时可以加速插入。
    - --no-create-info：不写重新创建每个转储表的 CREATE TABLE 语句。

 --default-character-set=latin1：按照原有的字符集导出所有数据，这样导出的文件中， 所有中文都是可见的，不会保存成乱码。

（4）打开 data.sql，将 SET NAMES latin1 修改成 SET NAMES gbk。

（5）使用新的字符集创建新的数据库。

create database databasename default charset gbk;

（6）创建表，执行 createtab.sql。

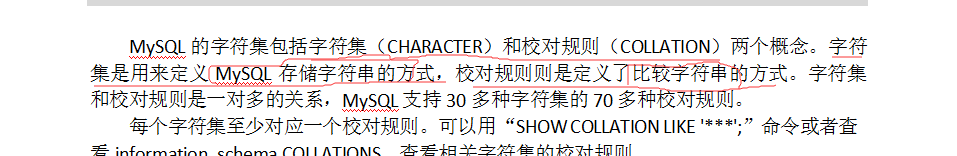
mysql -uroot -p databasename < createtab.sql

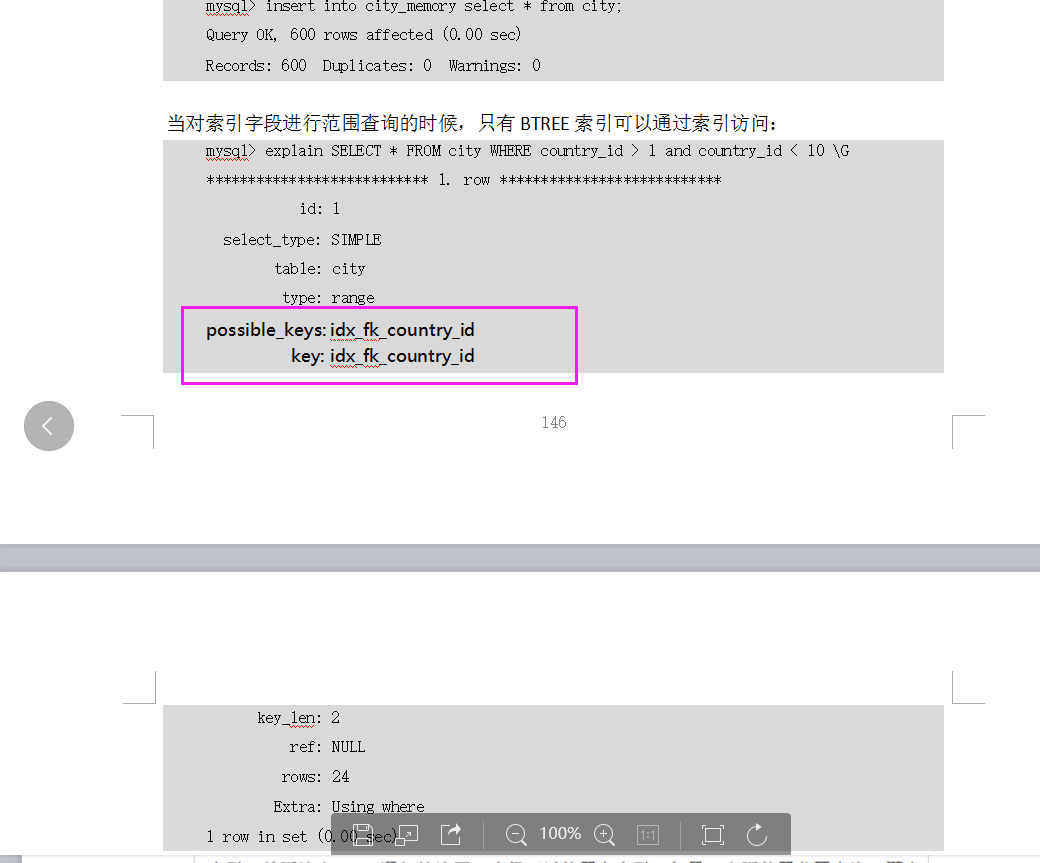
（7）导入数据，执行 data.sql。

mysql -uroot -p databasename < data.sql

# 索引

# 





https://www.cnblogs.com/hanybblog/p/6482981.html

http://www.cnitblog.com/aliyiyi08/archive/2008/09/09/48878.html