# 数据表操作

## 修改数据表

表单修改分为两部分：修改表本身和修改字段

**修改表本身：**修改表名和表选现象

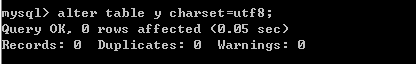
修改表名：alter table rename 老表名 [to] 新表名



修改表选项：修改字符集，校对集和存储引擎

alter table 表名 表选项 [=] 值;

修改字符集:



修改表的存储引擎：

alter table 表名 engine=引擎名

修改表的校对集:

**修改字段：**字段操作很多：新增，修改，重命名，删除。

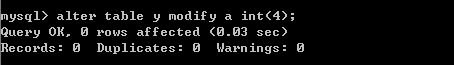
新增字段： alter table 表名 add [column] 字段名 数据类型 [列属性] [位置]

位置：新增字段可以放在表中的任意位置：first | after ;

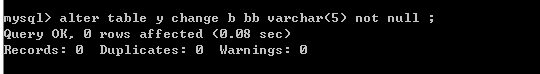


修改字段：通常修改字段属性或字段的数据类型。

alter table 表名 modify 字段名 数据类型 [属性] [位置] ;



重命名字段：alter table 表名 change旧字段名 新字段名 数据类型 [属性] [位置] ;



删除字段：alter table drop 字段名 ;



删除表的外键约束：

Alter table 表名 drop foreign key 外键约束名

删除数据表：

drop table [if exists]表1,表2,表3;



## 添加数据

## 修改数据

## 查看数据

## 外键

外键用来在两个表的数据之间建立链接，它可以是一列或者多列。一个表可以有一个或多个外键。外键对应的是参照完整性，一个表的外键可以为空值，若不为空，则每一个外键值必须等于另一个表中主键的某个值。

外键：首先它是表中的一个字段，它可以不是本表的主键，但对应另外一个表的主键。外键的作用是保证数据引用的完整性，定义外键后，不允许删除在另一个表中具有关联关系的行。外键的作用是保证数据的一致性和完整性。

主表（父表）：对于两个具有关联关系的表而言，相关联的字段中主键所在的哪个表即是主表。

从表（子表）：对于两个具有关联关系的表而言，相关联的字段中外键所在的哪个表即是主表。

语法：constraint 外键名 foreign key 字段名1[,字段名2,…….] references 主表名 主键列1[,主键列2,…..]

# 中文数据问题

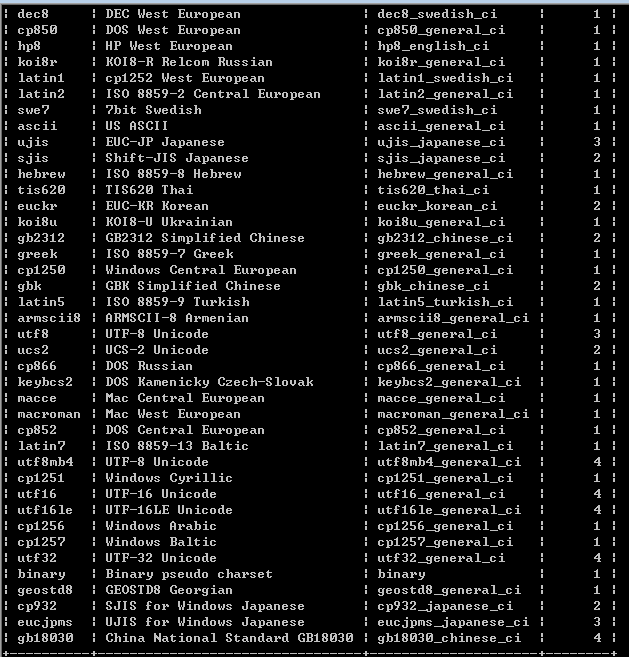
中文数据问题本质还是字符集问题

所有的数据库服务器认为（表现）的一些特性都是通过服务器的变量进行保存：系统先读取自己的变量，看看应该如何表现。

MySQL可以使用各种字符集来存储数据，包括单字符集合多字符集。

//查看服务器到底识别哪些字符集

show character set ;



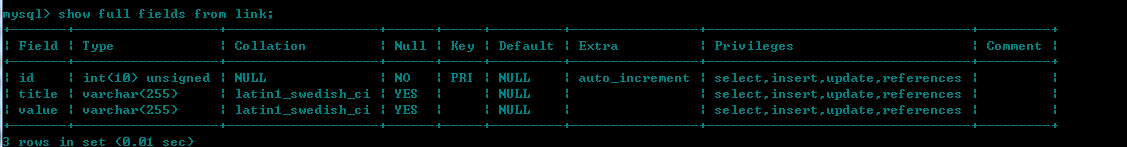
其中第4列maxlen大于1，那么该字符集为多字符集。

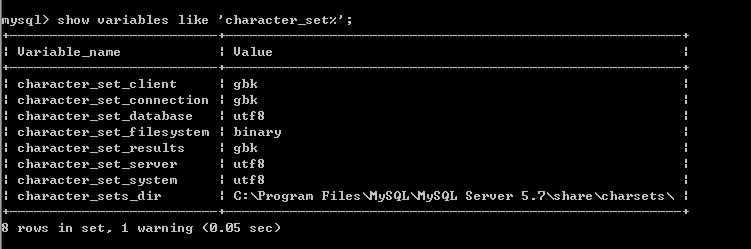
基本上服务器是万能的，什么字符集都支持

//既然服务器支持这么多 的字符集，总有一种是服务器默认的跟客户端进行打交道的字符集

show variables like ‘character\_set%’;

**PS:**





character\_set\_client:服务器默认的客户端来的数据字符集

character\_set\_connection:连接字符集

character\_set\_database:当前数据库的字符集

character\_set\_results:服务器默认的给外部数据的字符集

修改服务器默认的接受数据字符集：set character\_set\_client=新字符集

set 变量=值 ：修改的只是会话级别（当前客户端，当前连接有效，关闭无效）

设置服务器对客户端的字符集认识：可以使用快捷方式：set names 设置字符集。如：set names utf8;

# Unicode和UTF8区别

Unicode：是一种在计算机上使用的字符编码，它为每种语言中的每个字符设定了统一且唯一的二进制编码，以满足跨语言和跨平台进行文本转换和处理的要求，注意，Unicode是字符编码，不是字符集。

Unicode是基于通用字符集（Universal Character Set）的标准进行发展的，同时以书本的方式对外发表。

Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案，Unicode数字0～0x10FFFF来映射这些字符，最多可以容纳1114112个字符，或者说有1114112个码位。码位就是可以分配个字符的数字。

# 校对集问题

校对集：指对指定字符集下不同字符的比较规则，其特征如下：

1. 两个不同的字符集不能有相同的排序规则
2. 每个字符集有一个默认的排序规则。
3. 有一些常用的命名规则。

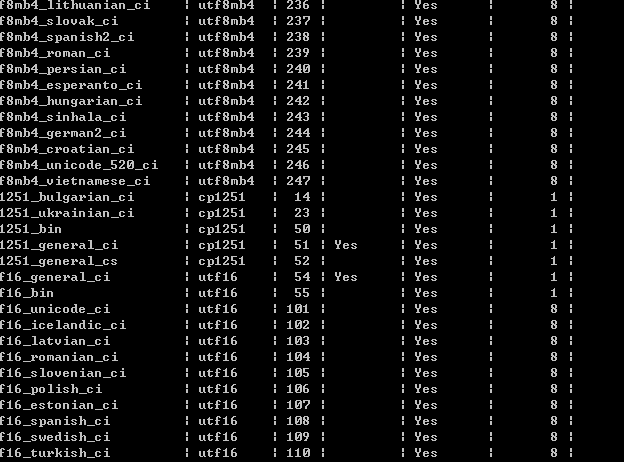
校对集有三种方式：

\_bin:binary,二进制比较，取出二进制位，一位一位的进行比较，区分大小写。

\_cs:case sensitive,大小写敏感，区分大小写。

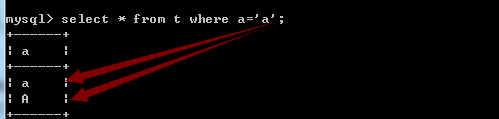
\_ci:case insensitive,大小写不敏感，不区分大小写

查看数据支持的所用的校对集：show collation



校对集应用：只有当数据进行比较的时候，校对集才生效。

校对集必须在没有数据之前声明好，如果有了数据，那么再进行数据集修改，那么修改无效。

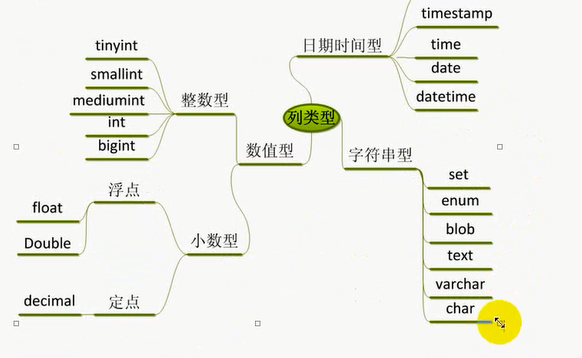


该

# 数据类型

所谓的数据类型：对数据进行统一的分类，从系统的角度出发为了能过统一的方式进行管理，更好的利用的有限的数据空间。

SQL中数据类型分为三类：数值类型，字符串类型和日期类型



显示宽度：没有特别的意义，只是默认的告诉用户可以显示的形式而已，实际上用户可以控制的，这种控制不会改变数据本身的大小。

显示宽度的意义：当数据不够显示宽度的时候，会自动让数据变成对应的显示宽度，通常需要搭配一个前导0来增加宽度，不改变值大小：zerofill（零填充）：零填充会导致数值自动变成无符号。

## 整数型



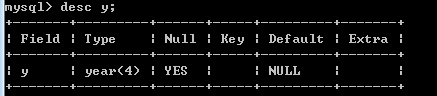
## 浮点型



## 时间类型

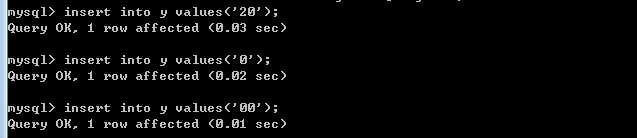


### YEAR



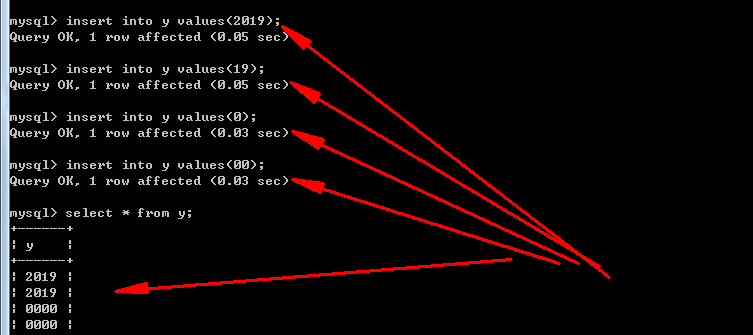
1. year以四位字符串或四位数字格式表示的YEAR，范围为1901～2155
2. 以两位字符串格式表示的YEAR，范围为’00～99’。’00～69’范围的值转换为

2000～2069。’70～99’范围的值转换为1970～1999。

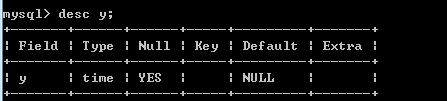


1. 以两位数字格式表示的YEAR，范围为1～99。1～69范围的值转换为

2001～2069。70～99范围的值转换为1970～1999。而0值转换为0000



### TIME



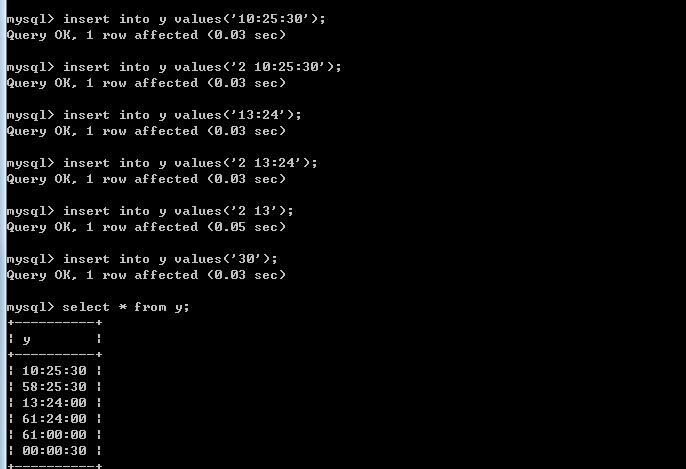
存储是需要3个字节，格式为”HH:MM:SS”，time类型的取值范围为：-838:59:59～838:59:59

**小时部分如此大的原因使TIME类型不仅可以表示一天的时间，还可以是某个事件过去的时间或两个事件的时间间隔。**

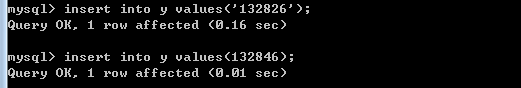
使用格式：

1．’D HH:MM:SS’格式的字符串，非严格格式语法：’HH:MM:SS’，’HH:MM’，’D HH:MM’，

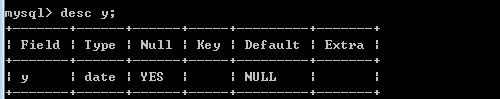
‘D HH’或’SS’。D表示日，可以去0-34之间的值。



2.’HHMMSS’格式，没有间隔符的字符串或者HHMMSS格式的数值。

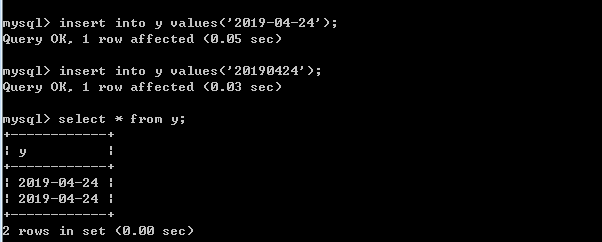


### DATE

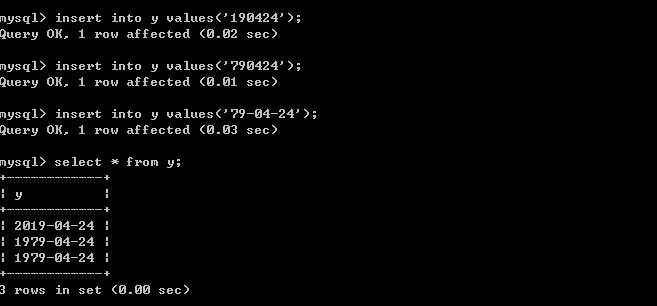


存储时需要3个字节，日期格式为：”YYYY-MM-DD”，在给DATE类型赋值时可以是字符串也可以是数字类型。

格式：1.’YYYY-MM-DD’或者’YYYYMMDD’字符串格式表示日期。

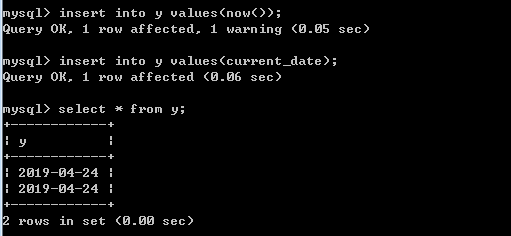


2.’YY-MM-DD’或者’YYMMDD’，其中YY表示两位的年值，MySQL使用以下解释：’00～69’范围的年值转换为’2000～2069’， ’70～99’范围的年值转换为’1970～1999’



3.YY-MM-DD或者YYMMDD数字格式表示日期。

4.使用CURRENT\_DATE或者NOW()插入当前系统时间。



### DATETIME

存储时需要8个字节。日期格式为’YYYY-MM-DD HH:MM:SS’，给DATETIME格式赋值时可以是字符串也可以是数字类型。

格式：1.’YYYY-MM-DD HH:MM:SS’或者’YYYYMMDDHHMMSS’字符串格式，取值范围为：’1000-01-01 00:00:00’～’9999-12-03 23:59:59’。

2.’YY-MM-DD HH:MM:SS’或者’YYMMDDHHMMSS’字符串格式。

3.YYYYMMDDHHMMSS或者YYMMDDHHMMSS数字格式表示日期和时间。

### TIMESTAMP

显示格式与DATETIME相同，显示宽度固定在19个字符，日期格式为”YYYY-MM-DD HH:MM:SS”，存储时需要4个字节。但是TIMESTAMP取值范围小于DATETIME。

## 字符串类型

### 定长字符串

定长字符串：char 磁盘在定义结构的时候，就已经确定了最终数据存储长度

char(L):L代表长度，可以存储的长度，单位为字符，最大长度值为255

### 变长字符串

变长字符串：varchar，在分配空间的时候，按照最大的空间分配，但实际用了多少是根据具体的数据来确定的。

varchar(L):L表示字符长度，理论长度是65536个字符，但是会多出1到2个的字节来确定存储的实际长度。

例如：varchar(10):的确存储10个汉字，utf8情况下10\*3+1=31个字节。

### 文本字符串

如果存储的数据量很大，通常来说超过255个字符就会使用文本字符串

文本字符串根据存储的数据格式进行分类:text和blob

Text:存储文字

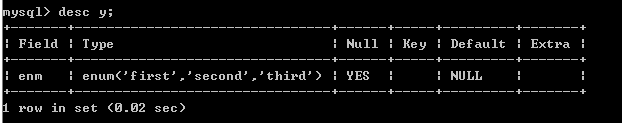
Blob:存储二进制数据（通常不用）

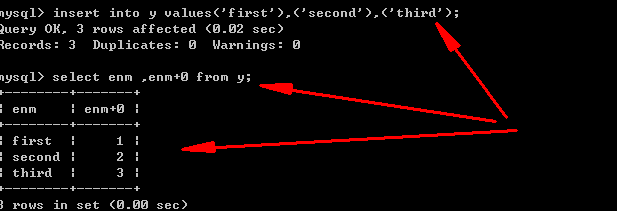
### 枚举字符串



枚举原理：枚举在进行数据规范时候（定义的时候），系统会自动建立一个数字与枚举元素的对应关系（关系放在日志中），然后在进行数据插入的时候，系统自动的将字符串转化中对应的数字存储，在进行数据查询的时候，系统自动的将数字转换成对应的字符串显示。

ENUM值在内部使用整数表示，每个枚举均有一个索引值：列表值所允许的成员值从1开始编号。

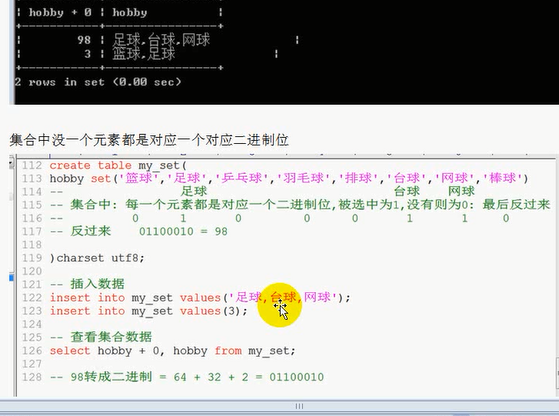




ENUM列总有一个默认值。如果将ENUM列声明为NULL，NULL值则为该列的一个有效值，并且默认值为NULL。如果ENUM列被声明为NOT NULL，其默认值为允许的值列表的第一个元素。

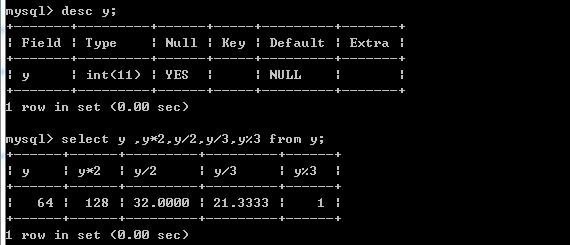
### 集合字符串

SET是一个字符串对象，可以有零或多个值，SET列最多可以有64个成员

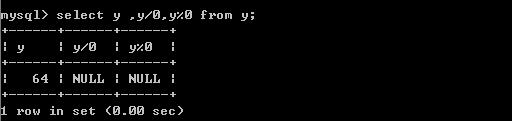


# 运算法

## 算术运算符



MySQL对无法被整除的数，会保存小数点后面四位。



除数为0的除法是没有意义的，如果被0除，则返回NULL。

## 比较运算符

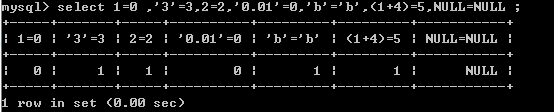
|  |  |
| --- | --- |
| 运算符 | 作用 |
| = |  |
| <=> | 安全的等于 |
| <>（!=） | 不等于 |
| <= |  |
| >= |  |
| > |  |
| IS NULL | 判断一个值是否为NULL |
| IS NOT NULL | 判断一个值不为NULL |
| LEAST | 有两个或多个参数时，返回最小值 |
| GREATEST | 当有2或多个参数时，返回最大值 |
| BEWTEEN AND | 判断一个值是否落在两个值之间 |
| ISNULL | 与 IS NULL作用相同 |
| IN | 判断一个值是IN列表中的任意一个值 |
| NOT IN | 判断一个值不是IN列表中的任意一个值 |
| LIKE | 通配符匹配 |
| REGEXP | 正则表达式 |

数值比较规则：（1）若有一个或两个参数为NULL，则比较运算的结果为NULL.

（2）若同一个比较运算中的两个参数都是字符串，则按照字符串进行比较

（3）若两个参数均为整数，则按照整数比较。

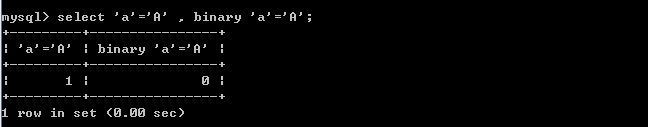
（4）若一个字符串和数字进行相等判断，则MySQL可以自动将字符串转换为数字。

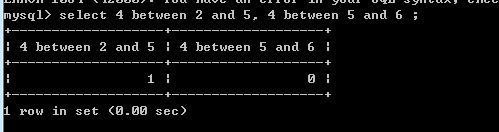


**MySQL中如何执行区分大小写的字符串比较？**

在window平台下，MySQL是不区分大小写的，因此字符串比较函数也不区分大小写，如果想执行区分大小写的

比较，可以在字符串前面添加BINARY关键字。

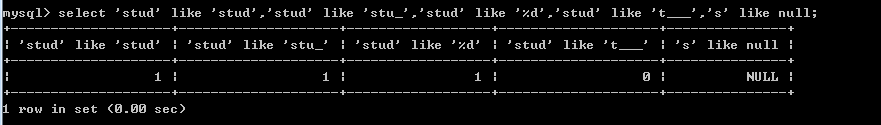




LIKE运算符在进行匹配时，可以使用两种通配符：

（1）’%’，匹配任何数目的字符，甚至包括零字符

（2）’\_’，只能匹配一个字符。



REGEXP运算符用来匹配字符串，语法格式为：expr REGEXP 匹配条件

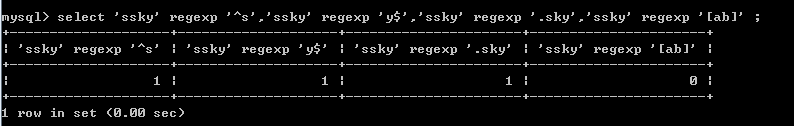
（1）’^’匹配以该字符后面的字符开头的字符串。

（2）’$’匹配以该字符后面的字符结尾的字符串。

（3）’.’匹配任何一个单字符。

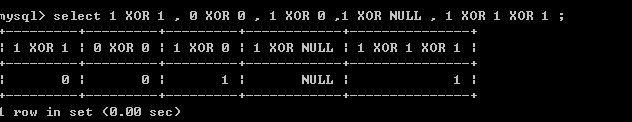
（4）’\*’匹配零个或多个在它前面的字符。

（5）’[…]’匹配在方括号内的任何字符。



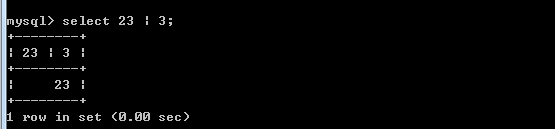
## 逻辑运算符

|  |  |
| --- | --- |
| 运算符 | 作用 |
| NOT 或者 ！ | 逻辑非 |
| AND 或者 && | 逻辑与 |
| OR 或者 || | 逻辑或 |
| XOR | 逻辑异或 |



## 位运算符

位或运算符（|）：按对应的二进制数逐位进行逻辑或运算。对应的二进制有一个或两个为1，则位的运算结果为1



23：10111

3：00011

10111

位与运算符（&）：按对应的二进制数逐位进行逻辑与运算，对应的二进制位都为1，则该为的运算结果为1，否者为0。

23：10111

3：00011

00011

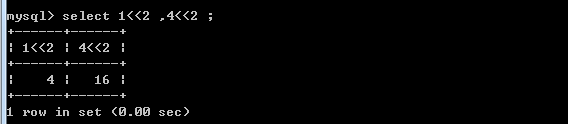
位异或运算符（^）：按对应的二进制数逐位进行逻辑异或运算，对应的二进制数不同时，对应为的结果才为1，如果两个对应位数都为0或者都为1，则对应为的结果为0。

23：10111

3：00011

10100

位左移运算符（<<）：使指定的二进制值的所有位都左移指定的位数。左移指定的位数后，左边高位的数值将被移出并抛弃，右边低位空出的位置用0补充。



技巧：左移的结果 result=num\*2^n ;

4=1\*2^2 16=4\*2^2;

1:0001 <<2 0100=4

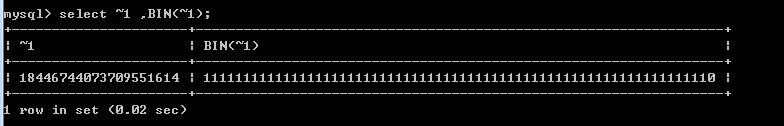
4:0100 <<2 10000

位右移运算符（>>）；使指定的二进制的所有位都右移指定的位数。右移指定位数之后，右移指定位数之后，右边低位的数值将被移除并丢弃，左边高位空出的位置用0补充。



1:0001 >> 1 0000 16：10000>>2 00100

位取反运算符（~）：按对应的二进制数逐位反转，即1取反后变0，0取反后变为1



# 函数

## 数学函数

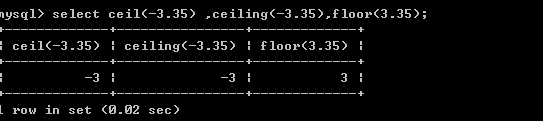
1. 绝对值函数：ABS(x)

2．圆周率函数PI()

3．平方根函数SQRT(x)

4．求余函数MOD(x,y)

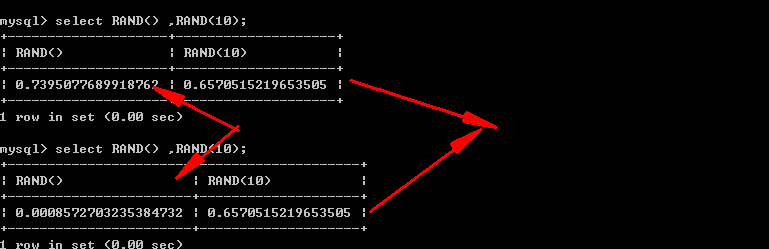
5．获取整数函数CEIL(x)，CEILING(x)和FLOOR(x);



CEIL(x)和CEILING(x)意义相同，返回不小于x的最小整数值，返回值转化一个BIGINT。

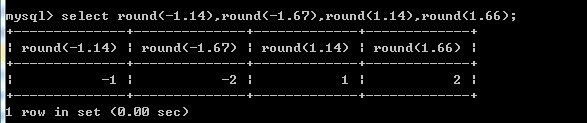
FLOOR()返回一个不大于x的最大整数值。

6．获取随机数函数RAND()和RAND(x)

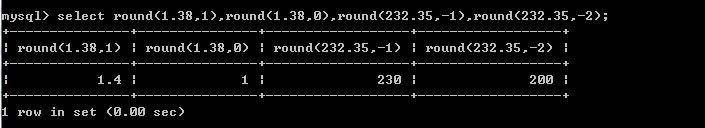


RAND(x)返回一个随机浮点数v，范围在0到1之间（即0<=v>=10），若指定x值，则它作为种子值，用来产生重复序列。

7、ROUND(x)返回最接近于参数x 的整数，对x值进行四舍五入。



ROUND(x,y)对操作数进行四舍五入操作，结果保留小数点后面指定的y位。若y为负值，则保留x值到小数点左边y位。



TRUNCATE(x,y)返回被舍去至小数点后y位的数字x。如y值为0，则结果不带有小数点或不带有小数点部分。若y设为负值，则截去x小数点左起第y位开始后面所有低位值。

注意：ROUND(x,y)在截去时会进行四舍五入，而TRUNCATE(x,y)在截去时并不会进行四舍五入。

8．符号函数SIGN(x)，返回参数的符号（-1,0和1）

9．幂运算函数POW(x,y)，POWER(x,y)，EXP(x)

10．对数运算函数LOG(x)和LOG10(x)

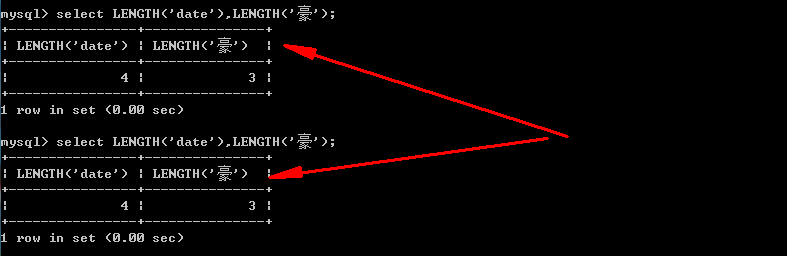
11．正弦函数SIN(x)和反弦函数ASIN(x)

12．余弦函数COS(x)和反余弦函数ACOS(x)

13．正切函数，反正切函数和与余正切函数TAN(x)

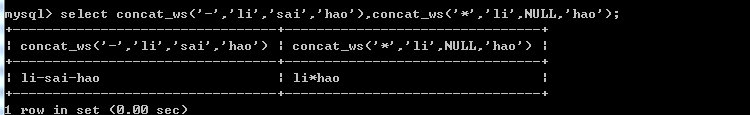
## 字符串函数

* CHAR\_LENGTH(str)返回字符串str包含的字符个数。
* LENGTH(str)返回字符串的长度



在UTF-8编码字符集时，一个汉字是3个字节，一个数字或字母是一个字节。

* CONCAT(str1,str2,….)合并字符串，如果有一个参数为NULL，则返回NULL
* CONCAT\_WS(x,str1,str2,…)是CONCAT的特殊形式，x是其他参数的分隔符，分隔符的位置必须放在要连接的两个字符串之间。分隔符可以是一个字符串，也可以是其他参数，如果分隔符为NULL，则结果为NULL。函数会忽略任何分隔符参数后面的NULL值。



* INSERT(s1,x,len ,s2)替换字符串函数，返回字符串s1，其子字符串起始于x位置和被字符串s2取代的len字符。如果x超过字符串长度,则返回值为原始字符串。如len的长度大于其他字符串的长度，则从位置x开始替换。若任何一个参数为NULL，则返回值为NULL。
* LOWER(str)或LCASE(str)将字符串str中的字母字符全部转换成小写字母。
* UPPER(str)或者UCASE(str)将字符串str中的字母字符全部转换成大写字母。
* LEFT(s,n)获取字符串s最左边n个字符
* RIGHT(s,n)获取字符串s最右边n个字符。
* LTRIM(s)字符串左侧空格字符串被删除
* RTRIM(s)字符串右侧空格字符串被删除
* TRIM(s)删除字符串两侧空格。
* TRIM(s1 FROM s)删除字符串s中两端所有的字符串s1，s1可选，在未指定的情况下，删除空格。
* REPEAT(s,n)返回一个重复的字符串s组成的字符串，字符串s的数目等于n。若n<=0，则返回一个空字符串，若s或n为NULL，则返回一个NULL。
* SPACE(n)返回一个由n个空格组成的字符串。
* REPLACE(s,s1,s2)使用字符串s2替换字符串s中所有的字符串s1.
* STRCMP(s1,s2)字符串大小比较函数，若所有的字符串均相同，则返回0；若根据当前分类次序，第一个参数小于第二个，则返回-1，其他情况返回1。该函数是大小写不敏感的
* SUBSTRING(s,n,len)获取子串，从字符串s返回一个长度为len的字符串，起始位置为n。

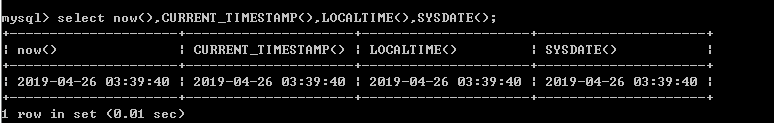
len省略，则返回n之后的所有字符。若n为负值，则字符串的位置起始于字符串结尾的n字符。

* LOCATE(str1,str)，POSITION(str1 IN str)和INSTR(str,str1)函数作用相同，返回子字符串str1在字符串str中的开始位置。
* REVERSE(s)字符串逆序函数
* ELT(N,str1,str2,str3,…..)如N=1，返回str1，N=2，返回str2，依次类推。若N小于1或大于参数的数目，则返回值为NULL。
* FIND\_IN\_SET(s1,s2)返回子串的位置。
* MAKE\_SET(x,s1,s2,……….)选取字符串的函数。

## 日期和时间函数

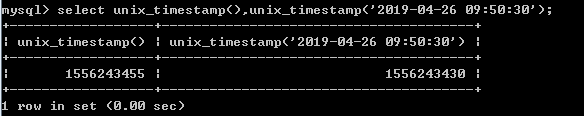
### 日期函数

* CURRENT\_DATE()和CURDATE()函数作用相同，将当前日期按照”YYYY-MM-DD”或”YYYYMMDD”格式返回。
* CURTIME()和CURRENT\_TIME()函数作用相同，将当前时间以”HH:MM:SS”或”HHMMSS”格式返回。
* CURRENT\_TIMESTAMP()，LOCALTIME()，NOW()和SYSDATE()函数作用相同，返回当前日期和时间。

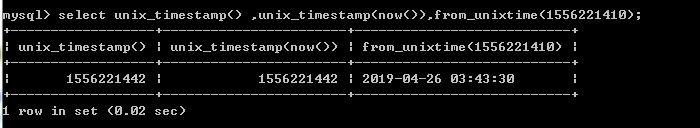


1. NOW()：返回执行SQL语句时的时间。
2. CURRENT\_TIMESTAMP()：NOW()相同。
3. SYSDATE()：该函数返回的是执行到当前函数时的时间。

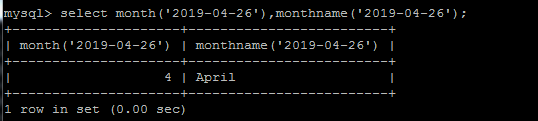
* UNIX\_TIMESTAMP(data)若无参数调用，则返回一个Unix时间戳（’1970-01-01 00:00:00’GMT之后的秒数）作为无符号整数。。



* FROM\_UNIXTIME(date)把UNIX时间戳转化为普通格式的时间。

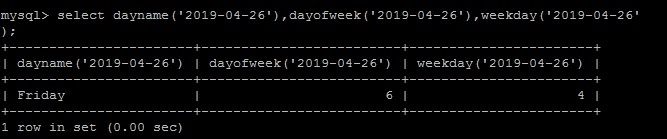


* UTC\_DATE()返回当前UTC（世界标准时间）日期值，格式为”YYYY-MM-DD”或”YYYYMMDD”
* UTC\_TIME()返回当前UTC时间值，格式为”HH:MM:SS”或”HHMMSS”
* MONTH(date)和MONTHNAME(date)返回月份，返回月份对应的英文全名



### 星期函数

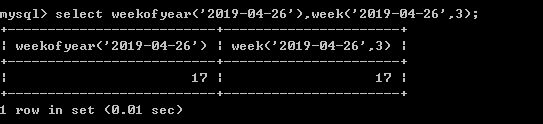
* DAYNAME(d)返回d对应的工作日的英文名称
* DAYOFWEEK(d)返回d对应的一周中的索引。1表示周日，2表示周一，3表示周二，……7表示周六。
* WEEKDAY(d)返回d对应的工作日索引。0表示周一，1表示周二，….，6表示周日。



* WEEK(d)计算日期d是一年中的第几周。WEEK()的双参数形式允许指定该星期是否起始于周日或周一，以及返回值的范围是否为从0～53或1～53。若Mode参数被省略，则使用default\_week\_format系统自变量。

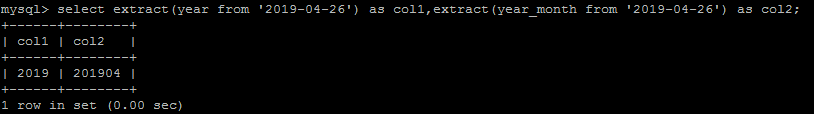
|  |  |  |  |
| --- | --- | --- | --- |
| Mode | 一周的第一天 | 范围 | Week1为第一周 |
| 0 | 周日 | 0～53 |  |
| 1 | 周一 | 0～53 |  |
| 2 | 周日 | 1～53 |  |
| 3 | 周一 | 1～53 |  |
| 4 | 周日 | 0～53 |  |
| 5 | 周一 | 0～53 |  |
| 6 | 周日 | 1～53 |  |
| 7 | 周一 | 1～53 |  |

* WEEKOFYEAR(d)计算某天位于一年中的第几周，范围是从1～53.。相当于WEEK(d,3)



### 时间函数

* DAYOFYEAR(date)返回date是一年中的第几天，范围从1～366。
* DAYONMONTH(date)返回date是一个月中的第几天，范围是1～31。
* YEAR(date)返回date对应的年份，范围是1970～2069
* QUARTER(date)返回date对应的一年中的季节值，范围是1～4。
* MINUTE(time)返回time对应的分钟数，范围从0～59
* SECOND(time)返回time对应的秒值，范围从0～59
* EXTRACT(type FROM date)函数所使用的时间间隔类型说明符同DATE\_ADD()或者DATE\_SUB()的相同，但它从日期中提取一部分，而不是执行日期计算。



* TIME\_TO\_SEC(time)返回已转化为秒的time值。
* SEC\_TO\_TIME(seconds)返回被转化为小时，分钟和秒数的seconds参数值，其格式为”HH:MM:SS”或”HHMMSS”

### 计算日期和时间的函数

DATE\_ADD(date,INTERVAL expr type)和DATE\_ADD(date ,INTERVAL expr type)作用相同

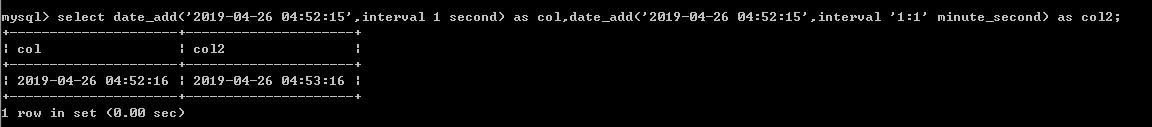
DATE\_SUB(date,INTERVAL expr type)和SUBDATE(date,INTERVAL expr type)作用相同

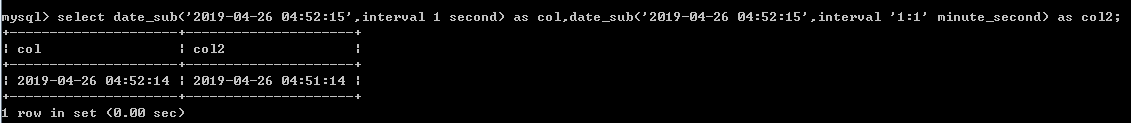
其中，date是一个DATETIME或TIME值，用来指定起始时间。

expr是一个表达式，用来指定从起始日期添加或减去的时间间隔值。expr是一个字符串；对于负值的时间间隔，它可以以一个减号’-’开头。

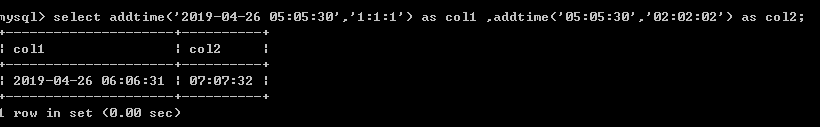
type为关键词，它指示了表达式被解释的方式。

|  |  |
| --- | --- |
| type值 | 预期的expr格式 |
| MICROSECOND | MICROSECONDS |
| SECOND | SECONDS |
| MINUTE | MINUTES |
| HOUR | HOURS |
| DAY | DAYS |
| WEEK | WEEKS |
| MONTH | MONTHS |
| QUARTER | QUARTERS |
| YEAR | YEARS |
| SECOND\_MICROSECOND | ‘SECONDS>MICROSECONDS’ |
| MINUTE\_MICROSECOND |  |
| MINUTE\_SECOND |  |
| HOUR\_MICROSECOND |  |
| HOUR\_SECOND |  |
| HOUR\_MINUTE |  |
| DAY\_MICROSECOND |  |
| DAY\_SECOND |  |
| DAY\_MINUTE |  |
| DAY\_HOUR |  |
| YEAR\_MONTH |  |

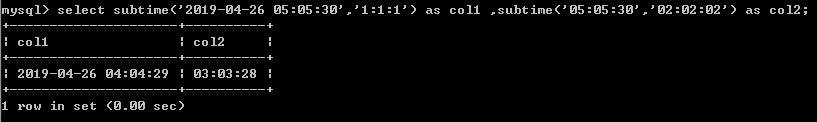




ADDTIME(date,expr)函数将expr值添加到date，并返回修改后的值，date是一个日期或者日期时间表达式，而expr是一个时间表达式。



SUBTIME(date,expr)函数将date减去expr值，并返回修改后的值，date是一个日期或者日期时间表达式，而expr是一个时间表达式。



DATEDIFF(date1,date2)返回起始时间date1和结束时间date2之间的天数。返回date1-date2

### 格式化函数

DATE\_FORMAT(date,format)根据给定的format格式显示date值。

TIME\_FORMAT(time ,format)根据给定的format格式显示time值。

%a：工作日的缩写名称。

%b：月份的缩写名称。

%d：该月日期，数字形式（00..31）

%H：以2位数表示24小时（00…23）

%i：分钟，数字形式（00…59）

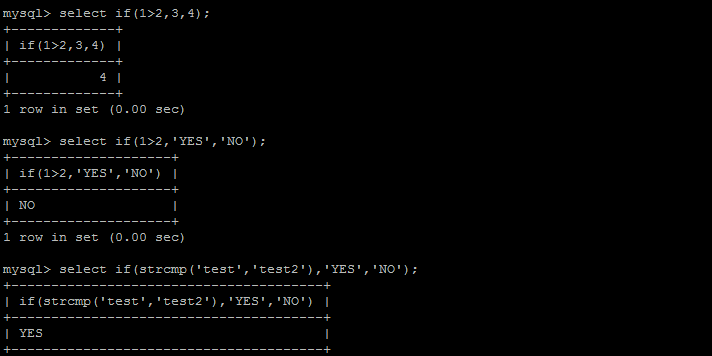
%M：月份名称

%p：上午(AM)，下午（PM）

%Y：4位形式表示年份。

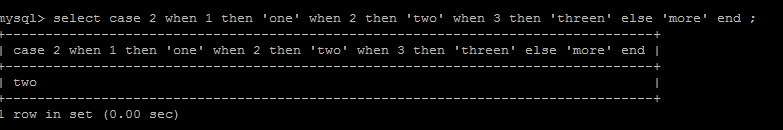
## 条件判断函数

* IF(expr ,v1,v2)如果表示expr是TRUE（expr<>0 and expr<>NULL），则IF()的返回值为v1，否则返回值为v2，该函数返回值可以是数字，也可以是字符串。



* IFNULL(v1,v2)假如v1不为NULL，则返回值为v1，否则返回值为v2。该函数的返回值可以是数字或字符串。
* CASE expr WHEN v1 THEN r1 [WHEN v2 THEN r2] [ELSE rn] END

该函数表示，如果expr值等于某个vn，则返回对应位置THEN后面的结果。如果与所有的值都不相等，则返回ELSE后面的rn。

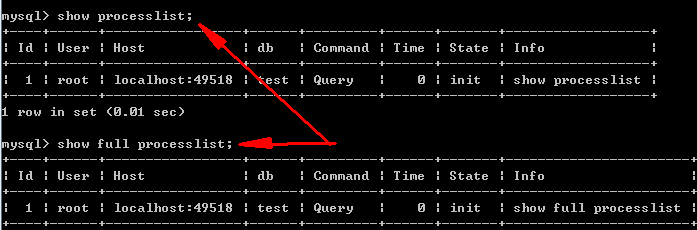


CASE WHEN v1 THEN r1 [WHEN v2 THEN r2 ] END

该函数表示，某个vn值为true时，返回对应位置then后面的结果，如果所有值都不为true，则返回else后的rn。

## 系统信息函数

* VERSION()返回指示MySQ服务器版本的字符串。
* CONNECTION\_ID()返回MySQL服务器当前连接的次数，每个连接都有各自的唯一ID



Id：用户登录MySQL时，系统分配的connection id

User：显示当前用户，如果不是root，这个命令就只显示用户权限范围内的sql语句。

Host：显示这个语句是从哪个ip的端口上发出的，可以用来追踪出现的问题语句的用户。

db：显示这个进程目前连接的是哪个数据库。

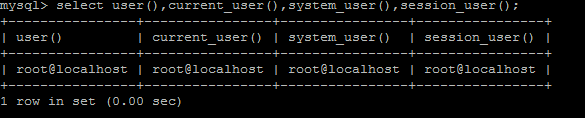
Command：显示当前连接的执行命令，一般取值是休眠（Sleep），查询（Query），连接（Connect）。

Time：显示这个状态持续的时间，单位是秒。

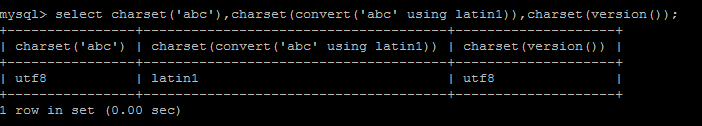
State：显示使用当前连接的sql语句的状态，

Info：显示这个sql语句，是判断问题语句的一个重要依据。

* DATABASE()和SCHEMA()函数返回使用utf8字符集的默认数据库名。
* USER(),，CURRENT\_USER()，SYSTEM\_USER()和SESSION\_USER()返回当前被MySQL服务器验证的用户名和主机名组合。



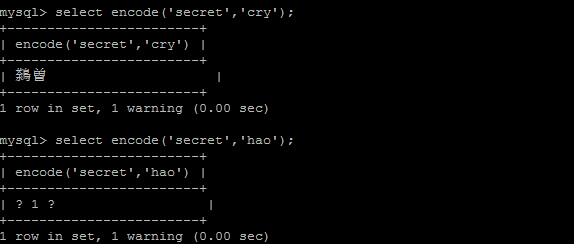
* CHARSET(str)返回字符串str自变量的字符集。



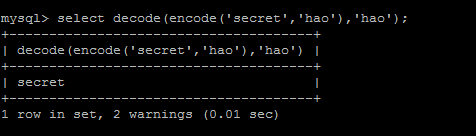
* COLLATION(str)返回字符串str的字符排列方式。
* LAST\_INSERT\_ID()自动返回最后一个INSERT或UPDATE为AUTO\_INCREMENT列设置的第一个发生的值。

## 加密函数

* PASSWORD(str)从原明文密码str计算并返回加密后的密码字符串，当参数为NULL时，返回NULL。
* MD5(str)为字符串计算出一个MD5 128比特校验和。该值以32位十六进制数字的二进制字符串并返回，如参数为NULL，则返回NULL。
* ENCODE(str,pswd\_str)使用pswd\_str作为面貌，加密str。使用DECODE()解密结果，结果是一个和str长度相同的二进制字符串。



* DECODE(crypt\_str,pswd\_str)使用pswd\_str作为密码，解密加密字符串crypt\_str.

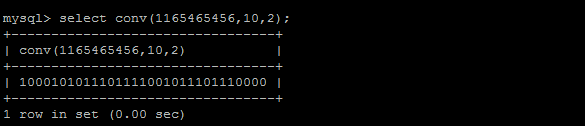


## 其他函数

1. 格式化函数FORMAT(x,n)

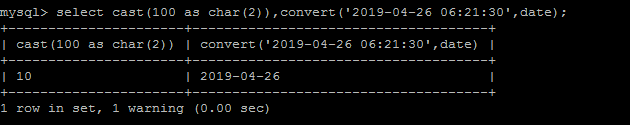
将数字x格式化，并以四舍五入的方式保留小数点后n位，结果以字符串的方式返回，若n为0，则返回结果函数不含小数部分。

1. 不同进制的数字进行转换函数CONV(N,from\_base,to\_base)



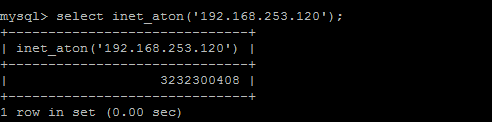
1. 改变数据类型的函数CAST(x AS type)和CONVERT(x,type)

可以转换的类型：BINARY，CHAR(n)，DATE,DATETIME,DECIMAL,SIGNED,UNSIGNED



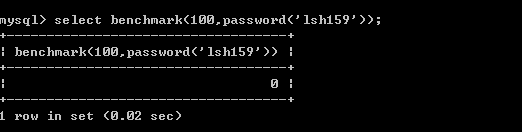
1. IP地址与数字相互转换函数INET\_ATON(expr)

给出一个作为字符串的网络地址的点地址表示，返回一个代表该地址数值的整数。



1. 重复执行指定操作的函数BENCHMARK(count,expr)

重复count次执行表达式expr，它可以用于计算MySQL处理表达式的速度。结果值通常是0（0只表示处理过程很快，并不是没有花费时间）。另一个作用是它可以在MySQL客户端内部报告语句执行的时间。



1. 改变字符集的函数CONVERT(str USING new\_charset)
2. 加锁函数和解锁函数GET\_LOCK(str,timeout) RELEASE\_LOCK(str) IS\_FREE\_LOCK(str)

IS\_USED\_LOCK(stsr)

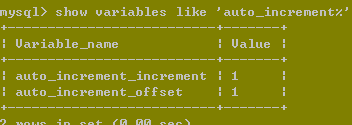
# 自动增长

自增长的特点：

1. 任何一个字段做自增长必须前提是本身是一个索引（key一栏有值）
2. 自增长字段必须是数字(整形)
3. 一张表中最多只有一个自增长

## 查看自增长变量

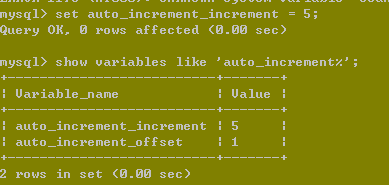
Show variables like ‘auto\_increment%’



## 修改自增长步长

修改是对整个数据的修改，而不是单张表（修改是回话级别）

Set auot\_increment\_increment = ?



## 删除自增长：自增长是字段的一个属性

Alter table 表名 modify 字段 属性；

# 唯一键

唯一键的本质与主键差不多：唯一键默认的允许自动为空，而且可以多个为空(空字段不参与唯一性比较)。

## 增加唯一键

* 在创建表的时候，在字段之后直接跟unique/unique key
* 在所有的字段之后增加unique key(字段列表):复合唯一键
* 在创建表之后添加唯一键

## 唯一键约束

唯一键与主键本质相同，唯一的区别就是唯一键默认允许为空，而且是多个为空。

## 更新唯一键&删除唯一键

更新唯一键：先删除后新增（唯一键可以有多个）

### 更新唯一键：

Alter table 表名 drop index 索引名字；--唯一键默认字段名为索引名

# 索引

索引：系统根据某种算法，将已有的数据(未来可能新增的数据)单独建议一个文件，文件能够实现快速上匹配数据，并且能够快速的找到对应表中的记录。

索引简介：索引是对数据库表中的一列或多列的值进行排序的一种结构，使用索引可提高数据库中特定数据的查询速度。索引是一个单独的，存储在磁盘上的数据库结构，它们包含着对数据表里所有记录的引用指针。使用索引用于快速找出在某个或多列中有一特定值的行，所有MySQL列类型都可以被索引。

索引是在存储引擎中实现的，因此，每种存储引擎的索引都不一定完全相同，并且每种存储

引擎也不一定支持所有索引类型。

存储引擎支持每个表至少16个索引，总索引长度至少为256字节。MySQL中索引的存储类

型有两种：BTREE和HASH，具体和表的存储引擎相关。MYISAM和InnoDB存储引擎只支持

BTREE索引；MEMORY/HEAP存储引擎可以支持HASH和BTREE索引。

索引优点：

1. 提高查询数据的效率。
2. 通过创建唯一索引，可以保证数据库中每行数据的唯一性
3. 在实现数据的参考完整性方面，可以加速表和表之间的连接
4. 在使用分组和排序子句进行数据查询时，可以显著减少查询中分组和排序时间。

索引缺点：

1．创建索引和维护索引要耗费时间，并且随着数据量的增加所耗费的时间也会增

加。

2．索引需要占用磁盘空间，处理数据表占数据空间之外，每一个索引还占用一定

的物理空间，如果大量的索引，索引文件可能比数据文件更快达到最大文件尺寸。

3．当对表中的数据进行增加，删除和修改时，索引也会动态的维护，这样就降低

了数据的维护速度。

如果某个字段需要作为查询条件经常使用，那么可以使用索引；

如果某个字段需要进行数据的有效性约束，也可以使用索引（主键，唯一键）

## 索引分类

Mysql中提供了多个索引

1. 主键索引：primary key
2. 唯一键索引：unique key
3. 全文索引：fulltext index
4. 普通索引：index

全文索引：针对文章内部的关键字进行索引。

### 普通索引和唯一索引

普通索引是MySQL中的基本索引类型，允许在定义索引的列中插入重复值和空值。

唯一索引，索引列必须唯一，但允许空值。如果是组合索引，则列值的组合必须唯一。主键索引是一种特殊的唯一索引，不允许空值。

### 单列索引和组合索引

单列索引即一个索引只包含单个列，一个表中可以有多个单列索引。

组合索引值在表的多个字段组合上创建索引，只有在查询条件中使用了这些字段的左边字段时，索引才会生效。使用组合索引时遵循最左前缀集合。

### 全文索引

全文索引FULLTEXT，在全文索引的列上支持值的全文索引，允许在这些索引列插入重复值和空值。全文索引可以在CHAR，VARCHAR或者TEXT类型的列上创建。MySQL中只有MyISAM存储引擎支持全文索引。

### 空间索引

空间索引对空间数据类型的字段建立的索引，MySQL中的空间数据类型有4中，分别是：GEOMETRY，POINT，LINESTRING和POLYGON。MySQL使用SPATIAL关键字进行扩展，使得能够用户创建正规索引类型的语句创建空间索引。创建空间索引的列，必须将其声明为NOT NULL，空间索引只能在存储引擎为MyISAM的表中创建。

## 创建索引

1. 创建表时创建索引

语法格式：

CREATE TABLE table\_name ( [col\_name data\_type] [UNIQUE | FULLTEXT | SPATIAL] [INDEX | KEY] [index\_name] (col\_name [length]) [ASC | DESC])

UNIQUE | FULLTEXT | SPATIAL：索引类型。

INDEX和KEY：两者作用相同，用来指定创建索引。

Index\_name：索引名称，为可选参数，如果不指定，默认为col\_name

col\_name：为需要创建索引的字段列，该列必须从数据表中定义的多个列中选择。

length：可选参数，表示索引的长度，只有字符串类型的字段才能制定索引长度。

ASC | DESC：制定升序或降序的索引值存储。

案例：创建普通索引

create table book(

bookid int not null ,

bookname varchar(266) not null,

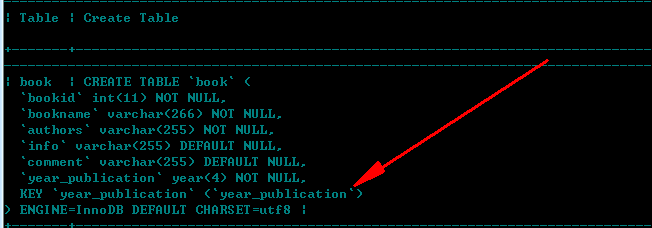
authors varchar(255) not null ,

info varchar(255) null ,

comment varchar(255) null ,

year\_publication year not null ,

index(year\_publication));



创建唯一索引：唯一索引的主要目的是减少查询索引列操作的执行时间。



创建单列索引：单列索引在表的某一个字段创建索引，一个表可以有多个单列索引。普通索引和唯一索引都为单列索引。

创建组合索引：在多个字段上创建索引。

create table t2(

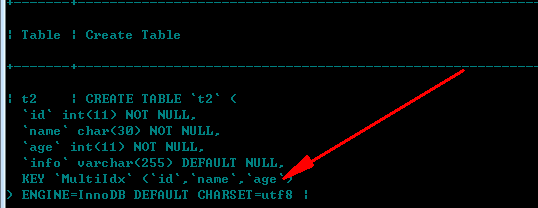
id int not null ,

name char(30) not null,

age int not null ,

info varchar(255),

index MultiIdx(id,name,age)

); 

创建全文索引：

create table t3(

id int not null ,

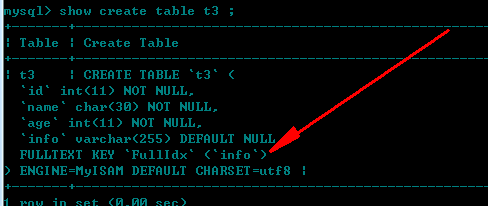
name char(30) not null,

age int not null,

info varchar(255),

fulltext index FullIdx(info)

)engine=MyISAM;



1. 在已经存在的表上创建索引

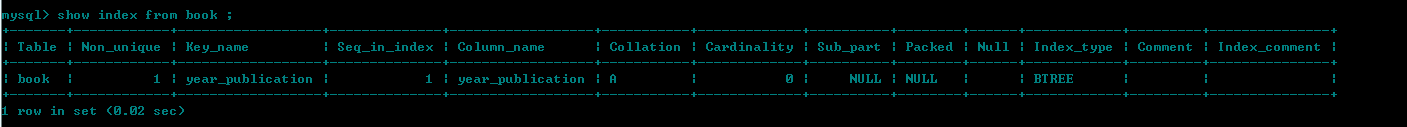
语法格式：

ALTER TABLE table\_name ADD [UNIQUE | FULLTEXT | SPATIAL] [INDEX | KEY] [index\_name]

(col\_name[length],….)[ASC | DESC]

alter table 表明 add index 索引名(表字段名or多个字段名)；

查看表的索引：show index from table\_naem ;



1. 使用CREATE INDEX创建索引

语法格式：

CREATE [UNIQUR | FULLTEXT | SPATIAL] INDEX index\_name On table\_name (col\_name[length],….) [ASC | DESC]

## 删除索引

1. 使用ALTER TABLE删除索引

ALTER TABLE table\_name DROP INDEX index\_name ;

2．drop index 索引名 on 表名；

## 查看索引

Show index from 表名；



Table：表的名称。  
Non\_unique：如果索引不能包括重复词（为唯一），则为0代表唯一索引。，则为1代表非唯一索引。  
 Key\_name：索引的名称。  
Seq\_in\_index：表示该字段在索引中的位置，单列索引该值从1开始，组合索引为每个字段

在索引中的定义的顺序。  
Column\_name：列名称。  
Collation：列以什么方式存储在索引中。在MySQL中，有值‘A’（升序）或NULL（无分类）。  
Cardinality： 索引中唯一值的数目的估计值。通过运行ANALYZE TABLE或myisamchk -a可以更新。基数根据被存储为整数的统计数据来计数，所以即使对于小型表，该值也没有必要是精确的。基数越大，当进行联合时，MySQL使用该索引的机会就越大。  
Sub\_part如果列只是被部分地编入索引，则为被编入索引的字符的数目。如果整列被编入索引，则为NULL。  
Packed　 指示关键字如何被压缩。如果没有被压缩，则为NULL。  
Null如果列含有NULL，则含有YES。如果没有，则该列含有NO。  
Index\_type　 用过的索引方法（BTREE, FULLTEXT, HASH, RTREE）。

# 关系

将实体与实体的关系，反应到最终数据库表的设计上来：讲关系分成三种：一对一，一对多，多对多。

## 一对一

一张表中数据一定只能与另外一张表的一条记录进行对应，反之亦然。

## 一对多

一张表中有一条记录可以对应另外一张表中的多条记录，但是反过来，另外一张表的一条记录只能对应第一张表的一条记录。

## 多对多

一张表中有一条记录可以对应另外一张表中的多条记录，同时，另外一张表的一条记录只能对应第一张表的多条记录。

# 范式

范式(normal format)：是一种离散数学中的知识，是为了解决一种数据的存储与优化的问题，保证数据的存储后，凡是能够通过关系寻找出来的数据，坚决不再重复存储，终极目标是为了减少数据的冗余。

范式：是一种分层结构的规范，分为六层，每一层都比上一层更加严格，若要满足下一层范式前提是满足上一层范式。

六层范式：1NF,2NF,3NF,…6NF. 1NF是最底层，要求最低，6NF是最高层，最严格。

## 第一范式(1NF)

在设计表存储数据时，如果表中设计的字段存储的数据，在取出来使用之前还需要额外的处理，那么说表的设计不满足第一范式，第一范式要求字段的数据具有原子性：不可再分。

## 第二范式(2NF)

第二范式：在数据表设计过程中，如何复合主键（多字段主键），且表中有字段并不是由整个主键来确定，而是依赖主键中的某个字段（主键的部分）:存在字段依赖主键的部分的问题，称之为部分依赖：第二范式就是解决表设计不允许出现部分依赖。

## 第三范式(3NF)

# 数据的高级操作

数据的操作：增删改查

## 新增数据

Insert into 表名[（字段名列表）] value（值列表）

数据的插入时候，假设主键对应的值已经存在了，插入一定会失败！

### 主键冲突

当主键存在冲突的时候(Duplicate key)可以选择性的进行处理：更新或替换

主键冲突：更新操作

Insert into 表名[(字段列表：包含主键)] value (值列表) on duplicate key update 字段=新值

主键冲突：替换操作

Replace into 表名 [(字段列表：包含主键)] values(值列表)

## 蠕虫复制

蠕虫复制：从已有的数据中去获取数据，然后将数据又进行新增操作：数据成倍的增加。

表创建高级操作：从已有的表创建新表(复制表结构)

create table 表名 like 数据库.表名

蠕虫复制：先查出数据，然后将查处的数据新增一遍

Insert into 表名[(字段列表)] select 字段列表/\* from 数据表名 ；、

蠕虫复制意义：

1. 从已有的表中拷贝数据到新表中
2. 可以迅速的让表中的数据膨胀到一定的数量级，测试表的压力以效率。

## 更新数据

update 表名 set 字段=值 where [条件]

高级新增语法

update 表名 set 字段= 值 [where条件] limit 条件。

## 删除数据

与更新类似，可以通过limit来限制数量

delete from 表名 【where条件】limit 条件。

删除：如果表中有自增长，那么删除后自增长不会还原。

思路：数据的删除是不会改变表结构，只能删除表后重建表。

truncate 表名;

## 查询数据

SELECT语句的基本格式：

SELECT {\* |<字段列表>}

[ FROM <表1>,<表2>…..

[WHERE<表达式> ]

[GROUP BY]

[HAVING]

[ORDER BY]

[LIMIT]

]

基本语法：select 字段列表/\* from 表名

完整语法：select [select 选项] 字段列表[字段别名]/\* from 数据源 [where条件语句][group by字句][having字句][order by字句][limit字句]

**select选项：**select对查出的结果的处理方式

All:默认的保留所有的结果

Distinct:去重，查出的数据，将重复给去除。

**字段别名：**当数据进行查询出来的时候，有时候名字并不一定满足需求(多表查询的时候，可能会有同名字段)需要对字段进行重命名：别名。

字段名 [as] 别名。

**数据源：**数据的来源，关系型数据库的来源都是数据表，本质上只有保证数据类似二维表，最终都可以作为数据源。

数据源分为多种：单表数据源，多表数据源，查询语句。

多表数据源 select \* from 表名1,表名2,….;

子查询：数据的来源是一条查询语句(查询语句的结果是二维表)

Select \* from（select 语句） as 表名

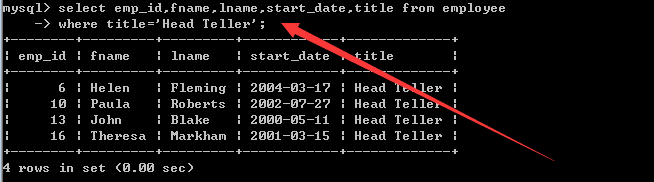
**where子句:**用来判断数据，筛选数据 where子句返回结果：0或1

where原理：where是唯一一个直接从磁盘获取数据的时候就开始判断条件，从磁盘取出一条记录，开始进行where判断：判断的结果如果成立就保存到内存，如果失败直接放弃。

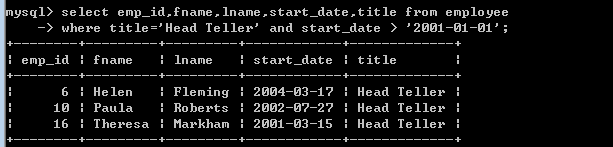
where过滤时，有两种过滤不被允许：

1. 由于数据还没有分组，因此现在还不能在where过滤条件中使用 where\_condition=MIN(col)这类对类型统计。
2. 由于没有进行列的选取操作，因此在where中使用列的别名是不允许的。

案例：查询employee表，获取头衔为Head Teller的雇员数据。



对查询进行扩展：它包含了第2个条件指明了只包含在2001年1月之后的入职的雇员。



**group by字句：**分组，根据某个字段进行分组(相同的放一组，不同的分到不同的组)

基本语法：group by 字段。

分组的意义：是为了统计数据(按组统计，按分组字段进行数据统计)

SQL提供了一系列的统计函数

count():统计分组后的记录数，每一组有多少记录

max():统计每组中最大是值。

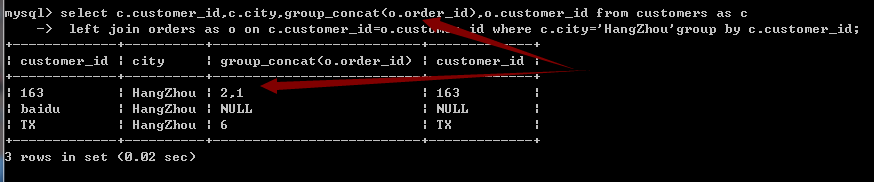
avg()：统计平均值。

sum()：统计和

分组会自动排序：根据分组字段默认升序。

多字段分组：先根据一个字段进行分组，然后对分组后的结果再次按照其他字段进行分组

group\_concat(字段名) 可以对分组的结果中的某个字段进行字符连接。



**隐式或显示分组：**

select max(avail\_balance) max\_balance ,

min(avail\_balance) min\_balance ,

avg(avail\_balance) avg\_balance,

sum(avail\_balance) tot\_balance,

count(\*) num\_accounts

from account where product\_cd='CHK';

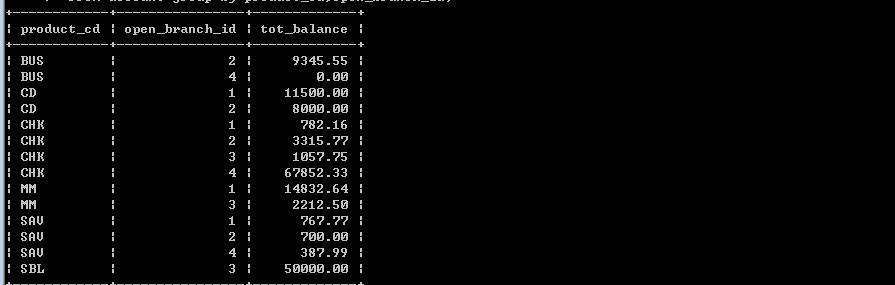
查询返回的每一个值都是聚集函数产生的，这些聚集函数作用于使用过滤条件product\_cd=’CHK’指定的分组上的所有行，这里并没有使用group by子句，因此它是一个隐式分组。

**回溯统计**：with rollup:任何一个分组后都会有一个小组，最后都需要向上级分组进行汇报统计：根据当前分组的字段这就是回溯统计，回溯统计的时候会将分组字段为空。

**案例：**根据产品和开户支行进行统计每种产品的余额。

select product\_cd,open\_branch\_id,sum(avail\_balance) tot\_balance

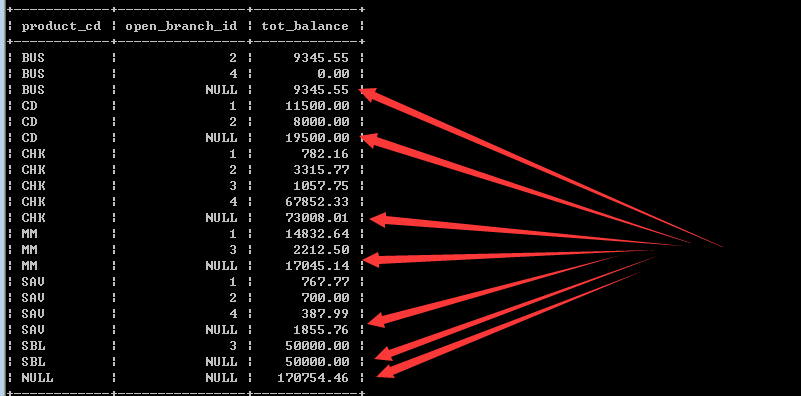
from account group by product\_cd,open\_branch\_id;



现在假设需要在为每种产品/支行组合计算合计余额的同时，还需要为每种产品单独计算合计树。

select product\_cd,open\_branch\_id,sum(avail\_balance) tot\_balance

from account group by product\_cd,open\_branch\_id with rollup;



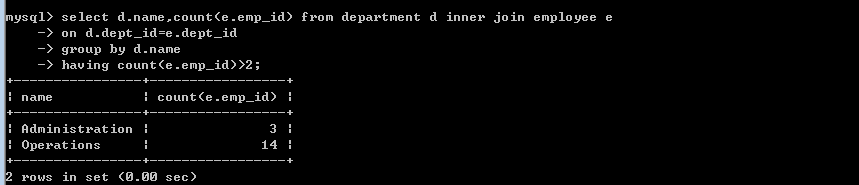
**多字段回溯：**考虑第一层分组会有回溯，第二次分组要看第一次分组的组数，组数是多少，回溯就是多少，然后再加上第一层回溯即可。

**having子句**：与where子句一样：进行条件判断的，having也是对分组进行过滤的筛选器。

where是针对磁盘数据进行判断，进入内存之后，会进行分组操作，返会结果就需要having来处理。

1. having几乎能做where所有的事情,where不能，比如对分组后的结果进行筛选。
2. having可以使用别名，where不能

案例：计算每个部门所含的雇员数，并返回至少包含2个雇员的部门



**order by**：根据某个字段进行升序或降序排序，依赖校对集

基本语法：order by 字段名 [asc|desc] –---asc升序 ，desc降序

排序可以进行对字段排序：先根据某个字段进行排序，然后排序好的内部，再按照某个数据进行再次排序。

案例：1.对account表查询

select open\_emp\_id,product\_cd from account;

select open\_emp\_id,product\_cd from account order by open\_emp\_id;

select open\_emp\_id,product\_cd from account order by open\_emp\_id,product\_cd;

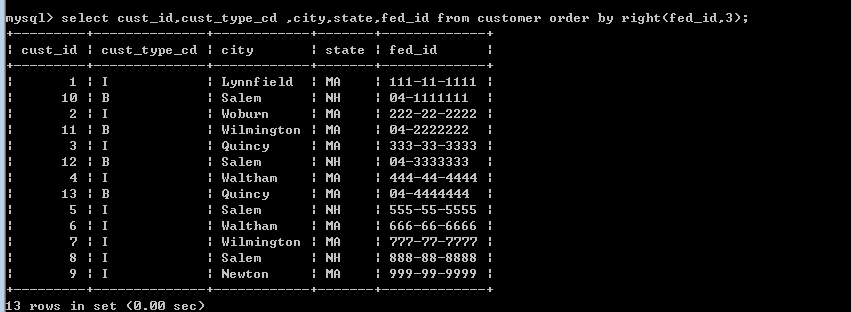
2．升序或降序排序：根据余额排序账户

select account\_id,product\_cd,open\_date,avail\_balance

from account order by avail\_balance desc;

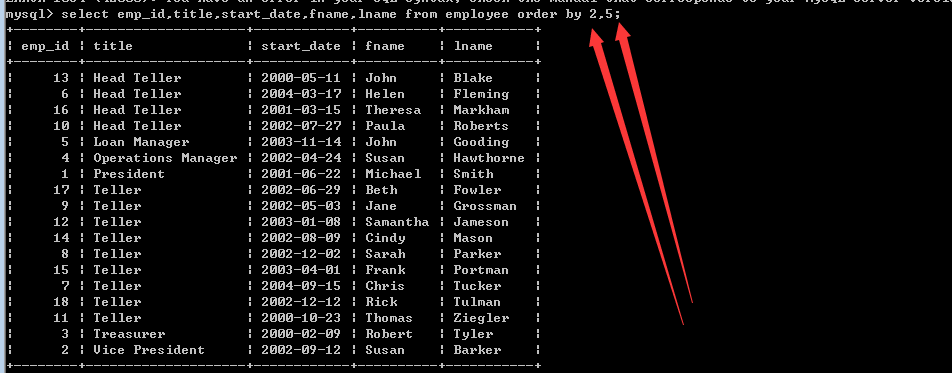
3．根据表达式排序：使用列数据对结果集进行排序十分有用，但是有时或许还需要根据一些并非存放在数据

库中，甚至可能没有在查询中出现的内容进行排序，而在orde by子句中增加表达式可以满足这种需求。如：customer表中，也许你会需要根据客户的联邦个人识别号码的最后3位数字进行排序。



4.根据数字占位符排序：如果需要根据select子句中的列来排序，那么可以选择使用该列位于select子句中个位

置号代替列明。



这种做法并不常见，因为如果select子句中的增加了新列，而没有同步改变order by子句中的序号，就可能导致预料之外的结果。

**limit子句：**是一种限制结果 的语句，限制数量。

Limit有两种使用方式：

方案一：

1. 只用来限制长度（数据量）:limit数据量
2. 限制起始位置，限制数量：limit起始位置，长度。

方案二：主要用于来实现数据的分页，为用户节省时间，提交服务器的响应效率，减少资源的浪费。

**Distinct:**如果在查询中指定了Distinct子句，则会创建一张内存临时表（如果内存中存在不下就放到磁盘上）。临时表的列增加了一个唯一索引，以此去除重复数据。

# 连接查询

连接查询：将多张表（可以大于2）进行记录的连接（按照某个指定的条件进行数据拼接）最终结果：记录数可能变化，字段数一定会增加(至少两张表的合并)

连接查询的意义：在用户查看数据的时候，需要显示的数据来自多张表

连接查询：join 使用方式：左表 join 右表

左表：在join关键字左边的表

右表：在join关键字右边的表

### 连接查询的分类

SQL中将连接查询分成四类：内连接，外连接，自然连接和交叉连接

create table customers

(

customer\_id varchar(10) not null ,

city varchar(10) not null,

primary key(customer\_id)

);

insert into customers select '163','HangZhou';

insert into customers select '9you','ShangHai';

insert into customers select 'TX','HangZhou';

insert into customers select 'baidu','HangZhou';

create table orders

(

order\_id int not null auto\_increment ,

customer\_id varchar(10) ,

primary key(order\_id)

);

insert into orders select null,'163';

insert into orders select null,'163';

insert into orders select null,'9you';

insert into orders select null,'9you';

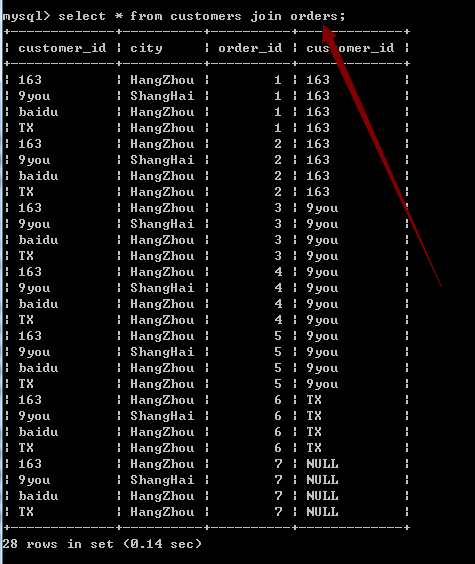
insert into orders select null,'9you';

insert into orders select null,'TX';

insert into orders select null,null;

#### 交叉连接

交叉连接：从一张表中循环取出每一条数据，每条记录都去另外一张表进行匹配：匹配一定保留，而连接本身字段就会增加(保留)，最终形成的结果叫做：笛卡尔积



案例：

select date\_add('2008-01-01',interval(ones.num+tens.num+hundreds.num ) day) dt from

(

select 0 num union all

select 1 num union all

select 2 num union all

select 3 num union all

select 4 num union all

select 5 num union all

select 6 num union all

select 7 num union all

select 8 num union all

select 9 num

) ones

cross join(

select 0 num union all

select 10 num union all

select 20 num union all

select 30 num union all

select 40 num union all

select 50 num union all

select 60 num union all

select 70 num union all

select 80 num union all

select 90 num

) tens

cross join(

select 0 num union all

select 100 num union all

select 200 num union all

select 300 num

) hundreds

where date\_add('2008-01-01',interval(ones.num+tens.num+hundreds.num) day)<'2009-01-01'

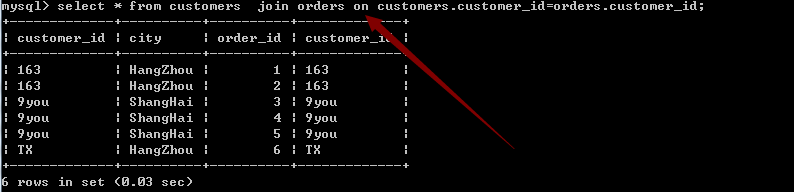
order by 1;

#### 内连接

内连接[inner]join:从左表中取出的每一条记录，去右表中与所有的记录进行匹配，匹配必须是某个条件在左表中与右表中相同，最终才会保留结果，否则不保留。

基本语法： 左表[inner] join 右表 on 左表.字段=右表.字段

on表示连接条件，条件字段就是代表相同的业务含义



内连接可以没有连接条件,没有on之后的内容，这个时候系统会保留所有的结果(笛卡尔积)

内连接可以使用where代替 on关键字(where没有on效率高)。

如果在一个连接查询中，涉及到的两张表都是同一个表，这种查询称为自连接查询。自连接查询是一种特殊的内连接，它是指相互连接的表在物理上为同一张表，但可以在逻辑上分为两张表。

**案例**：1.连接3个或更多的表：

2.将子查询结果作为查询表：查找所有Woburn支行中有经验的柜员所开设的账户

select a.account\_id ,a.cust\_id,a.open\_date,a.product\_cd

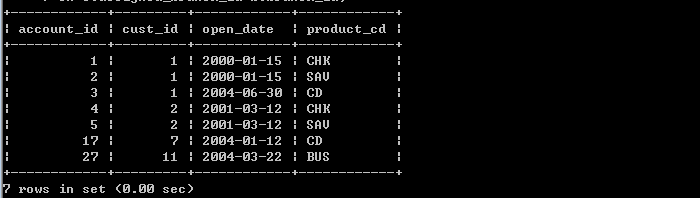
from account a inner join( select emp\_id,assigned\_branch\_id from employee

where start\_date<'2007-01-01' and (title='Teller' or title='Head Teller')) e

on a.open\_emp\_id=e.emp\_id

inner join(select branch\_id from branch where name='Woburn Branch') b

on e.assigned\_branch\_id=b.branch\_id;



第一个子查询的别名为e，它用于查询所有有经验的柜员。第二个子查询的别名为b，它用于查询Woburn支行的ID。

1. 连续两次使用同一个表：在连接多个表时，有时候可能需要多次连接同一个表。

select a.account\_id,e.emp\_id,b\_a.name open\_branch ,b\_e.name emp\_branch

from account a inner join branch b\_a

on a.open\_branch\_id=b\_a.branch\_id

inner join employee e

on a.open\_emp\_id=e.emp\_id

inner join branch b\_e

on e.assigned\_branch\_id=b\_e.branch\_id

where a.product\_cd='CHK';

4.自连接：不仅可以在同一个查询中多次包含同一个表，还可以对表自身进行连接。

#### 外连接

外连接(outer join):以某张表为主，取出里面的所有的记录，然后每条记录与另外一张表进行连接，不管能不能匹配上条件，最终都会保留，能匹配，正确保留，不能匹配，其他表的字段都置空Null.

外连接查询将查询多张表中相关联的行，内连接时，返回查询结果集中的仅是符合查询条件和连接条件的行，但有时候需要包含没有关联的行中数据，即返回查询结果集合中的不仅包含符合连接条件的行，而且还包括左表，右表或两个边接表中的所有数据行。

外连接分为两种：是以某张表为主，有主表

Left join:左外连接，以左表为主表。返回包括左表中的所有记录和右表中连接字段相等的

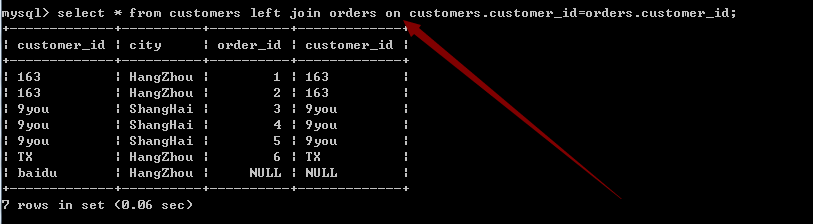
记录。

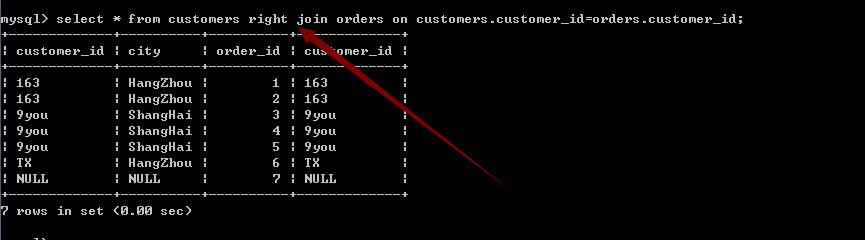
Right join:右外连接，以右表为主表。返回包括右表中的所有记录和左表中连接字段相等的

记录。

基本语法：左表 left/right join 右表 on 左表.字段=右表.字段

虽然左连接和右连接有主表差异，但是显示的结果：左表的数据在左边，右表的数据在右边





#### 自然连接

自然连接(natural join)：就是自动匹配连接条件：系统以字段名字作为匹配模式(同名字段就作为条件，多个同名字段都最为条件)

自然连接：可以分为自然内连接和自然外连接

自然内连接：左表 natural join 右表

### ANSI连接语法

SQL92版本的ANSI SQL标准。所有的主流数据库都采用了SQL92的连接语法。

select e.fname,e.lname,d.name from department d inner join employee e

on d.dept\_id=e.dept\_id;

该语句使用了内连接

select e.fname,e.lname,d.name from employee e,department d

where e.dept\_id=d.dept\_id;

这种旧方式并不包含on子句，而是在from子句中定义了各表的别名，并使用逗号隔开，然后在where子句中包含了连接条件。

ANSI语法优点：1.连接条件和过滤条件被分隔到两个子句中（on子句和where子句），使查询语句更具易于理解。

2.每两个表之间的连接条件都在它们自己的on子句中列出，这样不容易错误地忽略了某些连接条件。

3.使用SQL92连接语法的查询语句可以在各种数据库服务器中通用，而旧语法在不同的服务器上的表现可能略有不同。

案例：返回Woburn支行中所有熟练柜员（在2007年以前入职的柜员）开设的账户：

使用旧的连接方式；

select a.account\_id,a.cust\_id,a.open\_date,a.product\_cd

from account a,branch b,employee e

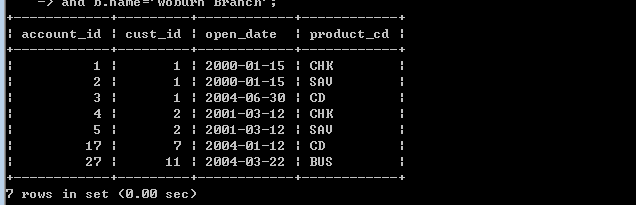
where a.open\_emp\_id=e.emp\_id

and e.start\_date<'2007-01-01'

and e.assigned\_branch\_id=b.branch\_id

and (e.title='Teller' or e.title='Head Teller')

and b.name='Woburn Branch';



使用这种查询，一是不太容易识别where子句中的条件哪些是连接条件，哪些是过滤条件；二是对于使用了何种连接类型也不是显而易见，此外还容易错误地遗漏某些连接条件。

使用ANSI连接语法：

select a.account\_id,a.cust\_id,a.open\_date,a.product\_cd

from account a inner join employee e

on a.open\_emp\_id=e.emp\_id

inner join branch b

on e.assigned\_branch\_id=b.branch\_id

where e.start\_date<'2007-01-01'

and (e.title='Teller' or e.title='Head Teller')

and b.name='Woburn Branch';

# 联合查询

当对两个数据集合执行集合操作时，必须首先应用下面的规范：

1. 两个数据集合必须具有同样数目的列；
2. 两个数据集中对应列的数据类型必须是一样的（或者服务器能够将其中一种转换为另外一种）。

## union操作符

联合查询：将多次查询(select语句)，在记录上进行拼接(字段不会增加)

基本语法：对条select语句构成：每一条select语句获取的字段数必须严格一致（但是字段类型无关）

select语句1 union[union选项] select 语句2….

union选项：与select选项一样有两种：

all:保留所有(不管重复)，得到的最终数据集的行数总是等于所要连接的各集合的行数之和

distinct:去重（整个重复）默认的

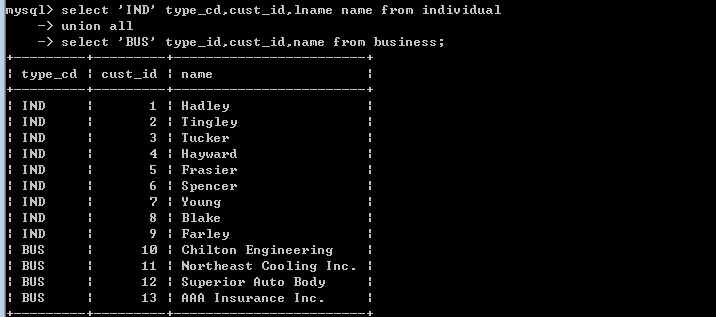
意义：分为两种：

1.查询同一张表，但是需求不同，如：查询一个班级男生升高 升序，女生 降序

2.多表查询，多张表的结构是完全一样的，保存的数据（结构）也是一样的。

在联合查询中 order by 不能直接使用，需要对查询语句使用().如果要order by 生效必须使用limit，limit使用限制的最大值。

案例：



该查询返回了13个客户，其中9行来自于individual表，而其他4行来自于bussiness表。

需要强调的是union all操作符并不删除重复项：

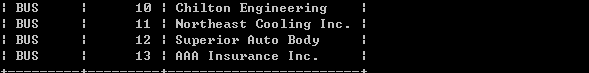
select 'IND' type\_cd,cust\_id,lname name from individual

union all

select 'BUS' type\_id,cust\_id,name from business

union all

select 'BUS' type\_id,cust\_id,name from business;



这个复合查询包含了3个select语句，其中两个是完全一样的。从结果上可以看出，来自于business表的4行数据被包含了两次。

## intersect操作符

ANSI的SQL规范定义了intersect操作符来执行**集合交**操作，不幸的是，MySQL6.0版还未实现intersect操作符，不过在Oracle或SQL Server2008中可以使用它。

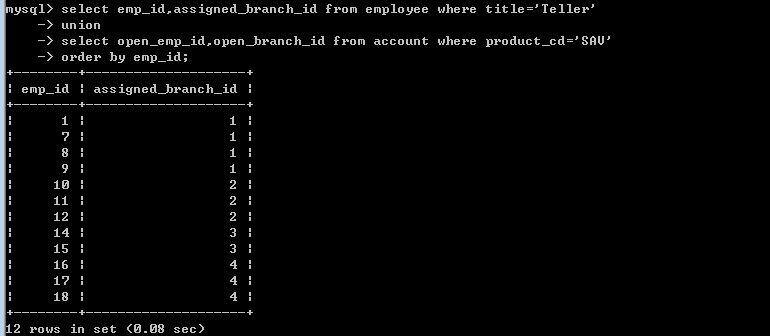
## except操作符

ANSI的SQL规范定义了except操作符来执行**集合差**操作，不幸的是，MySQL6.0版还未实现intersect操作符，不过在Oracle或SQL Server2008中可以使用它。

## 集合操作规则

### 对联合查询结果排序

如果需要对联合查询的结果进行排序，那么可以在最后一个查询后面使用order by子句。当在order by子句中指定要排序的列时，需要从联合查询的第一个查询中选择列明。通常情况下，联合查询中两个查询对应列的名字是相同的，但并不是强制的，如：



在这个两个查询指定的列名并不相同，如果在order by子句中指定的是来自第二个查询的列名，那么会发生错误。

### 集合操作符优先级

如果复合查询包含了两个以上使用不同集合操作符的查询，那么需要联合查询语句中确定查询执行的次序，以获取想要的结果。

select cust\_id from account where product\_cd in('SAV','MM')

union all

select a.cust\_id from account a inner join branch b

on a.open\_branch\_id=-b.branch\_id

where b.name='Woburn Branch'

union

select cust\_id from account where avail\_balance between 500 and 2500;



该联合查询包含了3个返回非唯一客户ID的查询，第一个和第二个查询使用union all操作符连接，而第二个和第三个查询之间使用的是union操作符。

将两个操作符的次序调换：

select cust\_id from account where product\_cd in('SAV','MM')

union

select a.cust\_id from account a inner join branch b

on a.open\_branch\_id=-b.branch\_id

where b.name='Woburn Branch'

union all

select cust\_id from account where avail\_balance between 500 and 2500;



从结果上看出，显然用不同的集合操作符构建联合查询会产生不同的查询结果。通常，联合查询包含3个或3个以上的查询语句，它们以自顶向下的顺序被解析和执行，但需要注意下面两点：

1. 根据ASSI SQL标准，在调用集合操作时，intersect操作符比其他操作符具有更高的优先级。
2. 可以使用圆括号对多个查询进行封装，以明确指定它们的执行顺序。

# 子查询

子查询：查询是在某个查询结果之上进行（一条select语句内容包含了另外一条select语句）

子查询结果作为外层另个一个查询的过滤条件，查询可以基于一个表或者多个表，子查询常用的操作符ANY（SOME），ALL，IN，EXISTS。

子查询返回如下类型的结果集：1.单列单行2.单列多行3.多列多行

子查询的好处：

（1）子查询允许结构化查询，这样就可以把一个查询语句的每一个部分隔开

（2）子查询提供了另一种方法来执行有些需要复杂的JOIN和UNION来实现的操作

一个子查询会返回一个标量，一个行，一个列或一个表，这些子查询被称为标量，列，行和表子查询。

子查询的分类：按位置分类，按结果分类

按位置分类：子查询(select语句)在外部查询(select语句)出现的位置

from子查询：子查询在from字后

where子查询：子查询在where条件中

exists子查询：子查询出现在exists里面

按结果分类：根据子查询得到的数据进行分类(理论上任何一个查询得到的结果都可以理解为二维表)

标量子查询：子查询得到的结果是一行一列

列子查询：子查询得到的结果是一列多行。

行子查询：子查询得好的结果是多列一行(多行多列)

上面几个出现的位置都是在where之后。

表子查询：子查询得到的结果是多行多列(出现的位置是在from之后)

按照期望值是数量，可以将子查询分为标量子查询和多值子查询。

若按照对外部查询的依赖可分为独立子查询和相关子查询。

标量子查询和多值子查询可以是独立子查询。

## 标量子查询

select account\_id,product\_cd,cust\_id,avail\_balance from account where open\_emp\_id <>

(select e.emp\_id from employee e inner join branch b on e.assigned\_branch\_id=b.branch\_id

where e.title='Head Teller' and b.city='Woburn');

## 列子查询

需求：查询所有在读班级的学生（班级表中存在的班级）

1. 确定数据源：学生表

Select \* from my\_student where c\_id in(？)

1. 确定有效的班级id:所有的班级id

Select id from my\_class ;

组合：select \* from my\_student where c\_id in (select id from my\_class);

列子查询返回的结果会比较，需要使用in作为条件匹配，其实在mysql中还有几个类似的条件：all ,some,any，in

**ANY：**any关键词必须与一个比较操作符一起使用，any关键词的意思是“对于子查询返回的列中的任意数值，如果比较结果为TRUE则返回TRUE”

Select s1 from t1 where s1 > any(select s1 from t2 );

**IN：**in是“=ANY”的别名

**SOME：**词语some是any的别名，使用some很少。

**ALL：**词语all必须与比较操作符一起使用，all的意思是“对于子查询返回的列中的所有值，如果比较结果为TRUE，则返回TRUE”

**EXISTS：**关键词后面是一个任意子查询，系统对子查询进行运算以判断它是否返回行，如果至少返回一行，那么EXISTS的结果为true，此时外层查询语句将进行查询；如果子查询没有返回任何行，那么EXISTS返回的结果为false，此时外层语句将不进行任何查询。

## 行子查询

行子查询：返回的结果可以是多行多列（一行多列），行子查询需要构建行元素:行元素是由多个字段构成。

需求：查询学生中，年龄最大和身高最高的学生。

1. 确定数据源

Select \* from my\_student where age=? And height=?

1. 确定最大年龄和最高身高

Select max(age),max(height) from my\_student;

-

行子查询组合：

Select \* from my\_student where

(age,height) = -- 行元素

(select max(age),max(height) from my\_student)

## 表子查询

表子查询：返回的结果是多行多列的二维表：子查询返回的结果当做二维表来使用。

需求：找出每一个班级最高的学生。

1. 确定数据源：现将学生按照身高进行降序排序

Select \* from my\_student order by height desc ;

1. 从每一个班级中选出第一个学生

Select \* from my\_student group by c\_id;

组合： select \* from (select \* from my\_student order by height desc) as student group by c\_id ;

表子查询：from子查询：得到的结果作为from的数据源。

## Exists子查询

Exists:是否存在的意思，exists子查询是用来判断某些条件是否满足（垮表），exists是接在where之后的，exists返回的结果只有1和0。

Exists是一个非常强大的谓词，它允许数据库高效地检查指定查询是否产生某些行，通常Exists的输入时一个子查询，并关联到外部查询，但这不是必须的。根据子查询是否返回行，该谓词返回TRUE或FALSE。与其他谓词和逻辑表达式不同的是，无论输入子查询是否返回行，Exists都不会返回UNKOWN。

# 视图

视图是一个虚拟表，是从数据库中一个或多个表中导出来的表，视图还可以从已经存在的视图的基础上定义。

视图一经定义便存储在数据库中

## 创建视图

CREATE [OR REPLACE] [ALGORITHM={UNDEFINED|MERGE|TEMPTABLE}] VIEW view\_name [(column\_list)] AS SELECT\_statement [WITH [CASCADED | LOCAL] CHECK OPTION]

REPLACE：替换已经创建的视图

ALGORITHM：视图选择的算法：UNDEFINED表示MySQL将自动选择算法，MERGE表示将使用的视图语句与视图定义合并起来，使得视图定义的某一部分取代语句对应的部分。TEMPTABLE表示将视图的结果存入临时表，然后将临时表来执行语句。

CASCADED：表示更新视图时要满足所有相关视图和表的条件，默认值。

LOCAL：表示更新视图时满足该视图本身定义的条件即可。

## 视图数据操作

1. 查看视图

* 使用DESCRIBE语句查看视图基本信息

describe view\_name ;

* 使用SHOW TABLE STATUE查看视图基本信息

Show table status like ‘view\_name’

* 使用SHOW CREATE VIEW语句查看视图

Show create view view\_name ;

* 在views表中查看视图详细信息

Select \* from information\_schema.views

1. 修改视图

* 使用CREATE OR REPLACE VIEW语句修改视图

CREATE [OR REPLACE] [ALGORITHM={UNDEFINED|MERGE|TEMPTABLE}] VIEW view\_name [(column\_list)] AS SELECT\_statement [WITH [CASCADED | LOCAL] CHECK OPTION]

* 使用ALTER语句修改视图

ALTER [ALGORITHM={UNDEFINED | MERGE | TEMPTABLE}] VIEW view\_name [(column\_list)] AS SELECT\_statement [WITH [CASCADED | LOCAL] CHECK OPTION]

1. 更新视图

### 新增数据

1. 多表视图是不能新增数据的
2. 可以向单表视图中新增数据，但是视图中包含的字段必须有基表中所有不能为空（或者没有默认值）字段

## 视图算法

视图算法：系统对视图以及外部查询视图的select 语句的一种解析方法。

视图算法定义为三种

Undefined:未定义（默认的），这不是一种实际使用是算法，是一种推卸责任的算法，告诉系统，视图没有定义算法，系统自己看着办。

Temptable:临时表算法，系统应先执行视图的select语句，后执行外部查询语句。

Merge:合并算法，系统应该将试图的select语句和外部查询语句进行合并，然后执行（效率高）。

算法指定：在创建视图的时候进行指定算法

Create algorithm=算法名 view 视图名 as select语句。

算法选择：如果试图的select语句中包括一个查询字句（五字句），而且很可能顺序比外部的查询语句都要靠后，一定是用算法temptable

# 事物

**事物的四个基本要求：**原子性（Atomicity），一致性（Consistency），隔离性（Isolation）和

持续性（Durablility）。

原子性：一个事物就是一个单元工作，当中可能包括数个步骤，这些步骤必须全部执行成功，若一个失败，则整个事物声明失败，事物中其他步骤必须撤销曾经执行过的动作，回到事物前的状态。

一致性：事物作用的数据集合在事物前后必须一致，若事物成功，真个数据集合都必须是事物操作后的状态，若事物失败，整个数据集合必须与开始事物前一样没有变更，不能发生整个数据集合部分变更，部分没有变更转态。

隔离性：在多人使用的情况下，每个用户可能进行自己的事物，事物与事物之间，必须互不干扰，用户不会意识到别的用户正在进行事物，就好像只有自己在进行操作一样。

持续性：事物一旦成功，所有的变更必须保存下来。

## 事物操作

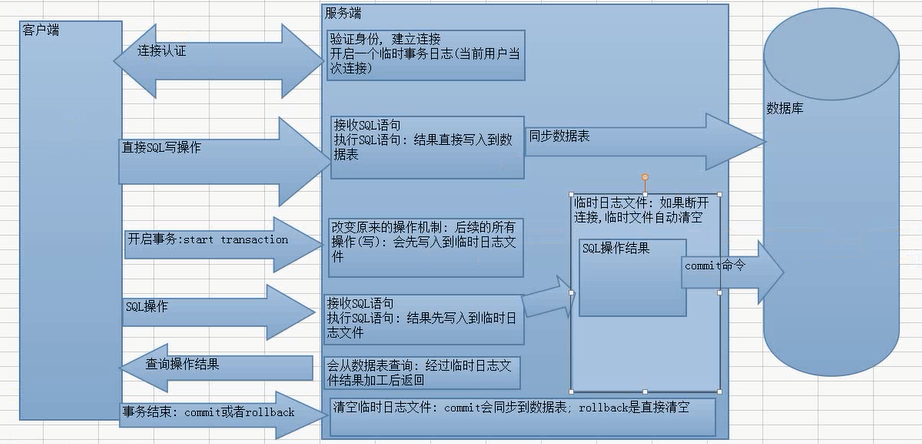
事物操作分为两种：自动事物（默认的），手动事物。

手动事物：操作流程

1. 开启事物：告诉系统一下所有的操作（写）不要直接写入到数据库中，先存放到事务日志中。Start transaction .
2. 进行实物操作。
3. 关闭事物：选择性的将日志文件中操作的结果保存到数据表中（同步）或者直接清空事物（原来操作全部清空）。
4. 提交事物：同步数据表（操作成功）commit
5. 回滚事物：直接清空日志表（）rollback

## 事物原理

事物操作原理：事物开启之后，所有的操作都会临时保存到事物日志中，事物日志只有得到commit命令才会同步到数据表，其他任何情况都会清空（rollback，断点，断开连接）

ck

## 回滚点

回滚点：在某个成功操作之后，后续的操作有可能成功有可能失败，但是不管成功还是失败，前面的操作已经成功，可以在当前成功的位置，设置一个点，可以供后续操作失败之后回到该位置，而不是返回所有的操作，这个点称之为回滚点。

设置回滚点语法：savepoint 回滚点名

回到回滚点语法：rollback to 回滚点名

# 变量

变量分为：系统变量和自定义变量

## 系统变量

系统定义好的变量，大部分的时候不需要使用系统变量：系统变量是用来控制服务器的表现。

查看系统变量：show variables;

查看具体变量：任何一个有数据返回的内容都是有select查看

select @@变量名。

### 修改系统变量

修改系统变量分为：会话级别和全局级别

会话级别：临时修改，当前客户端连接有效。

set 变量名 = 值 或者 set @@变量名=值

全局级别：一次修改，全局有效。

set globel 变量名=值

## 自定义变量

系统为了区分系统变量，规定用户自定义变量必须使用@符号

set @变量名 = 值 ；

MySQL允许从数据表中获取数据，然后赋值给变量：两种方法。

1. 边赋值边查看结果

select @变量名 := 字段 from 数据源

1. 只赋值不查看结果：要求很严格，数据记录最多只允许获取一条数据：mysql不支持数组

select 字段列表 from 数据源 where 条件into 变量列表

# 数据备份和还原

备份：将当前已有的数据或记录保存。

还原：将以保留的数据恢复到对应的表中。

数据的备份和还原有很多：数据表备份，单表数据备份，SQL备份，增量备份。

## 数据表备份

不需要通过SQL来备份，直接进入到数据库文件夹复制对应的表结果以及数据文件，以后还原的时候直接将备份的文件放入即可。

要想保持备份的一致性，备份前需要对相关表执行LOCK TABLES操作，然后对表执行FLUSH TABLES。这样当复制数据库目录中的文件时，允许其他用户继续查询表。需要FLUSH TABLES语句来确保开始备份前将所有激活的索引页写入硬盘。当然，也可以停止MySQL服务器再执行备份。

数据表备份有前提条件：根据不同的存储引擎有不同的要求。

Mysql的存储引擎：innodb和myisam等。

伏笔Innodb和Myisam数据存储方式：

Innodb:只有表结构,，数据全部存储到iddata1文件中

Myisam:表，数据和索引单独分开储存。

数据表备份适用于Myisam存储引擎，

## 单表备份

每次只能备份一张表，只能备份表数据（表结构不备份）

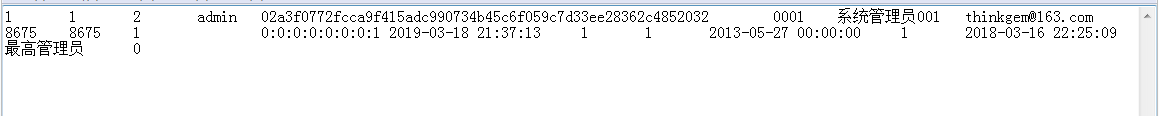
通常使用：将表中的数据导入到外部文件。

备份：从表中选出一部分数据保存到外部文件中（outfile）。

* 使用SELECT …INTO OUTFILE FILENAME导出文本文件

Select \* / 字段列表 into outfile ‘文件所在路径’ from 数据源





Select coulunlist from table where condition into outfile ‘filename’ [options]

高级备份：自己制定字段和行的处理方式。

Select \* /字段列表 into outfile ‘文件所在路径’ fields字段处理 lines行处理 from 数据源。

Field字段处理：

Enclosed by :字段使用什么包裹，默认使用’’空字符串。

Terminated by :字段以什么结束，默认使用’\t’。

Escaped by ：特殊字符使用什么处理，默认使用’\\’反斜杠处理。

Lines行处理：

Starting by :每行以什么开始，默认使用’’空字符串。

Terminated by:每行以什么结束，默认使用’\r\n’





Fields和lines都是可选的，如果都选了，则fields必须在lines之前。

* 使用mysqldump命令导出文本文件

MySQLdump创建一个包含创建表的CREATE TABLE语句的tabelname.sql文件，和一个包含其数据的tablename.sql文件。

语法格式：mysqldump –T Path –u username –p db\_name [table] [options]

Options选项：--fields-terminated-by=value：设置字段之间的分隔字符，可以为单个或多个字符，默认情况下为制表符。

--fields-enclosed-by=value：设置字符的包围字符

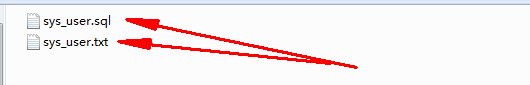
--fields-optionally-enclosed-by=value：设置字符的包围字符，只能为单个字符，只能包括CHAR和VERCHAR等字符数据字段

--fields-escaped-by=value：控制如何写入或读取特殊字符，只能为单个字符，即设置转义字符，默认值为反斜线。

--lines-terminated-by=value：设置每行数据结尾字符。







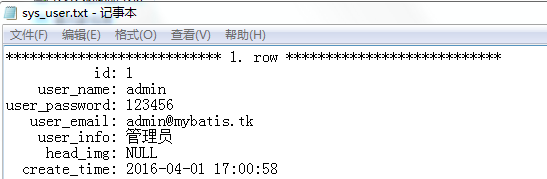
* 使用mysql命令导出文本文件

mysql –u root –p –exexute=”select 语句” > filename.txt:





将每条记录分为多行显示：





将数据记录到html中。

* 使用LOAD DATA INFILE方式导入文本文件

数据还原：将一个外部保存到的数据文件重新恢复到表中（数据库中表结构必须存在）。

load data infile ‘文件路径’ into table 表名[(字段列表) fields 字段处理 lines 行处理]。

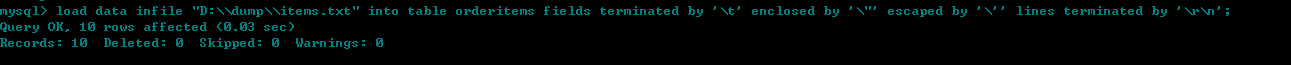
fields terminated by value

fields enclosed by value

fields escaped by value

fields starting by value

lines terminated by value



* 使用mysqlimport命令导入文本文件

语法格式：mysqlimport –u username –p db\_name filename.txt [options]

Options:--fields-termated-by=value

--fields-enclosed-by=value

--fields-optionally-enclosed-by=value

--fields-escaped-by=value

--lines-terminated-by=value

--ignore-lines=n：忽略数据文件的前n行。

## SQL备份

备份的是SQL语句：系统会对表结构和数据进行处理，变成对应的sql语句，然后进行备份，还原的时候只要执行sql语句即可。

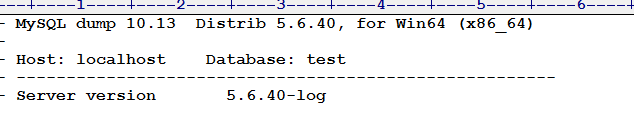
备份：mysql没有提供对应的语句，需要使用myql提供对应的软件：mysqldump.exe。

Mysqldump.exe是一种客户端，需要连接认证。

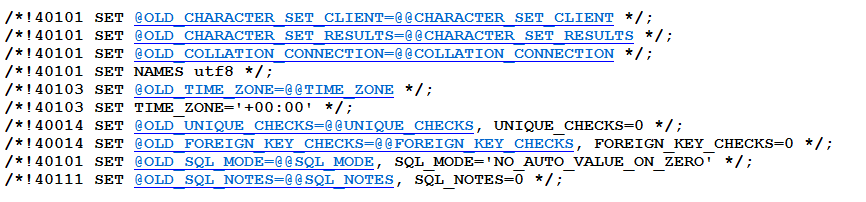
Mysqldump/mysqldum.exe –u –p 数据库名 [数据表名1，数据表名2…] > 外部文件路径（简易使用.sql结尾）。

* 使用mysqldump备份单个数据库中的所有表

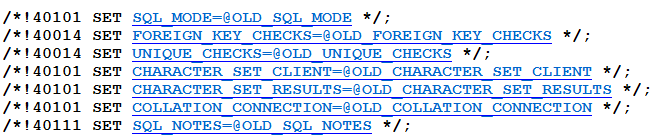




备份文件开头表明了使用MySQLdump工具的版本号，然后是备份账户的名称和主机信息，以及备份的数据库的名称，最后是MySQL服务器的版本号。



备份文件的set语句，这些语句将系统变量赋予了用户自定义的变量。以确保被恢复的数据库的系统变量和原理备份时的变量相同。



备份文件的最后几行，MySQL使用SET语句恢复服务器系统原来的值。

备份文件中的’--’字符开头的行为注释语句，以’/\*!’，’\*/’结尾的语句为可执行的MySQL注释。

* 使用mysqldump备份数据库中的某个表

mysqldump –u username –p password db\_name [table\_name [,table\_name]] > filename.sql



* 使用msqldump备份多个数据库

Mysql –u username –p password –databases [db\_name,[db\_name….]] > filename.sql



同样我们也可以使用—all-databases参数备份所有的数据库。

* Mysqldump其他选项参数

使用命令:mysqldump –help查看

* 使用mysqlhotcopy工具快速备份

MySQLhotcopy是一个Perl脚本，它使用LOCK TABLES，FLUSH TABLES和cp或scp来快速备份数据库。它是备份数据库或单个表的最快途径，但它只能运行在数据库目录所在的机器上，并且只能备份MyISAM类型的表。

语法格式：

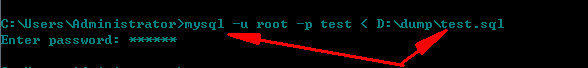
Mysqlhotcopy db\_name1,…..dbnamen filepath

### 数据还原

* 使用mysql.exe客户端还原

mysql.exe/mysql –u –p 数据库名 < 文件路径。

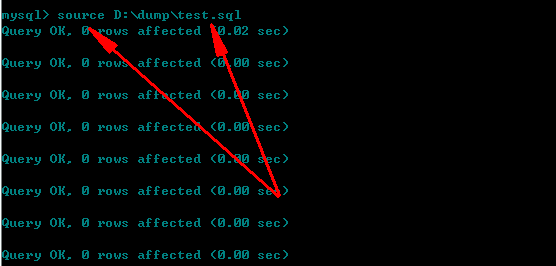
执行该语句前，必须先在mysql中创建数据库。



* 使用sql指令还原。

如果已经登录mysql，还可以使用source命令导入sql文件。

语法格式：source 备份文件路径。



* 使用mysqlhotcope快速恢复

在MySQL服务器停止运行时，将备份文件的数据库文件复制到MySQL存放数据的位置，重新启动MySQL服务即可

## 数据库迁移

* 相同版本的MySQL数据库之间的迁移

格式：mysqldump –h hostname1 –u username1 –p PWD1 db\_name | msql –h hostname2 –u username2 –pPwd2

* 不同版本的MySQL数据库之间的迁移
* 不同数据库之间的迁移

Window系统下，可以使用MyODBC实现MySQL和SQL SERVER之间的迁移

MySQL官方提供了MySQL Migration Tookit也可以在不同数据库之间进行数据迁移。

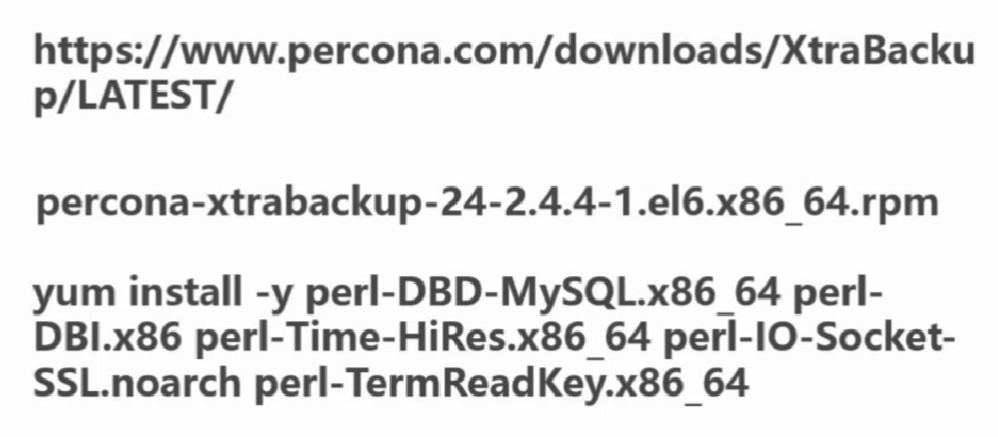
## 增量备份

增量备份是在上一次全量或增量备份的基础上，对于更改的数据进行备份。

MySQL提供的mysqldump并不支持增量备份，所以我们需要第三方工具。如：xtrabackup

Xtrabackup用于在线备份innodb存储引擎的表。

Innobackupex是对xtrabackup的封装并提供了MyISAM表的备份功能。



Mysqldump常用参数：

|  |  |
| --- | --- |
| 参数 | 含义 |
| -u, --user=name |  |
| -p, --password[=name] |  |
| -P, --port=# |  |
| -h, --host=name |  |
|  |  |
| --triggers |  |
| -R, --routines |  |
| -l, --lock-tables |  |
| -x, --lock-all-tables |  |
| --hex-blob |  |
| --master-data[=#] |  |

# 触发器

触发器（trigger）:事先为某张表绑定好一段代码，当表中的某些内容发生改变时，系统会自动触发代码执行。

触发类型：事件类型，触发时间，触发对象。

事件类型：增删改，三种类型：insert delete和update

触发时间：befor和after

触发对象：表中的每一条记录。

一张表最多只能有一种触发时间的一种类型：最多一张表有6个触发器

需求：有两张表，一张订单表，一张商品表，每生成一张订单是，库存就会自动减少。

## 创建触发器

在mysql高级结构中：没有大括号，都是使用对用的字符符号对应。

触发器的基本语法：

Delimiter 自定义符号:后续代码中只要碰到该符号才算结束

Create trigger 触发器名字 触发时间 事件类型 on 表名 for each row

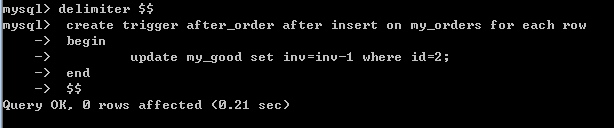
Begin

End

语句结束符：自定义符号

--将临时修改修正回来

Delimiter ;

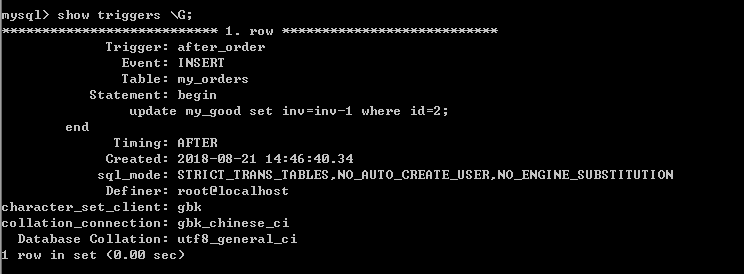


存在问题：

## 查看触发器

查看所有的触发器或者模糊匹配

Show triggers[like ‘pattern’]



查看触发器创建语句

Show create trigger 触发器名字；

所有的触发器都会保存到一张表中：information\_schema.trigger

## 修改触发器

触发器只能删除，不能修改，只能在删除后修改。

Drop trigger 触发起名

## 触发器记录

触发器记录：不管触发器是否出发了，只要当某种操作准备执行时，系统就会将当前要操作的记录的当前状态和即将执行之后新的状态给分别保留下来，供触发器使用，其中，要操作当前状态保存到old中，操作之后的可能形态保存给new。

Old:代表旧记录，new:代表新纪录。

删除的时候是没有new的，插入的时候是没有old。

# 存储过程和函数

存储过程如同一门程序设计语言，同样包含了数据类型、流程控制、输入和输出和它自己的函数库。

## 存储过程

1．创建存储过程

基本语法：

create procedure sp\_name([proc\_parameter])

[characteristic…] routine\_body  
begin  
.........  
end

sp\_name：存储过程名称

proc\_parameter：指定存储过程的参数列表，列表形式为：

[IN | OUT | INOUT] param\_name type

IN表示输入参数，OUT表示输出参数，INOUT表示既可以输入也可以输出。

param\_name表示参数名称，type表示参数类型，该类型可以是MySQL中的任意类型。

characteristic：指定存储过程的特性，有以下取值：

* LANGUAGE SQL：说明routine\_body部分是由SQL语句组成的，当前系统支持的语言为SQL，SQL是LANGUAGE特性的唯一值。
* [NOT]DETERMINISTIC：
* {CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA}：
* SQL SECURITY{DEFINER | INVOKER}：
* COMMENT “string”：注释信息

调用存储器：call sp\_name()  
注意：存储过程名称后面必须加括号，哪怕该存储过程没有参数传递。

delimiter //

create procedure Proc()

begin

select \* from fruits ;

end //

ps：”delimiter //”语句的作用是将MySQL的结束符设置为//，因为MySQL默认的语句结束符为分号;，为了避免与存储过程中SQL语句的结束符冲突，需要使用delimiter改变存储过程的结束符。存储过程定义完毕后，需要回复默认的结束符。

1. 调用存储过程

语法：CALL sp\_name([parameter[,….]])

delimiter //

create procedure CountProcl(in sid int ,out num int)

begin

select count(\*) into num from fruits where s\_id=sid;

end //

delimiter ;

调用：call CountProcl(101,@num);select @num ;

1. 查看存储过程

* 使用SHOW STATUS语句查看存储过程和函数状态

SHOW {PROCEDURE | FUNCTION} STATUS [like ‘pattern’]

如：show procedure status like ‘C%’ \G;

* 使用SHOW CREAET语句查看存储过程和函数定义

SHOW CREATE {PROCEDURE | FUNCTION} sp\_name

* 从information\_schema.Routines表中查看存储过程和函数信息

MySQL中 的所有存储过程和函数都存储在information\_schema数据库下的Routines表中。

SELECT \* FROM information\_schema.Routines where ROUTINE\_NAME=’sp\_name’

1. 修改存储过程

语法：ALTER PROCEDURE sp\_name [characteristic …]

sp\_name：存储过程的名称。

characteristic：指定存储函数的特性，可能取值为：

* CONTAINS SQL：表示子程序中包含SQL语句，但不包含读或写数据的语句
* NO SQL：表示子程序中不包含SQL
* READS SQL DATA：表示子程序中包含读数据的语句
* MODIFIES SQL DATA：表示子程序中包含写数据的语句
* SQL SECURITY {DEFINER | INVOKER}：指明谁有权限来执行
* DEFINER：表示只有定义者自己才能执行
* INVOKER：表示调用者可以执行
* COMMENT ‘string’表示注释信息

1. 删除存储过程

语法：DROP PROCEDURE [IF EXISTS] sp\_name ;

drop procedure sp\_name。

## 存储函数

1．创建存储函数

语法：

CREATE FUNCTION func\_name([func\_parameter])

RETURNS type

[characteristic …]routine\_body

func\_name：存储函数的名称

func\_parameter：存储过程的函数参数列表：[IN | OUT | INOUT] param\_name type

IN表示输入参数，OUT表示输出参数，INOUT表示既可以输入也可以输出。

param\_name表示参数名称，type表示参数类型，该类型可以是MySQL中的任意类型。

RETURNS type：表示函数返回的类型

delimiter //

create function NameByZip()

returns char(50)

return (select s\_name from suppliers wherer s\_call=’1452’);

//

delimiter ;

2．调用存储函数

调用存储函数和调用MySQL内部函数一样。

delimiter //

create function CountProc2(in sid int)

return int

begin

return (select count(\*) from fruits where s\_id=sid);

end //

delimiter ;

调用：select CountProc2(101);

1. 删除存储函数

语法：DROP FUNCTION [IF EXISTS] sp\_name ;

drop procedure sp\_name。

## 其他

1．变量的使用

定义变量：

declare var\_name [,varname]… data\_type [default value]；

如：declare myparam int default 100 ;

为变量赋值：

SET var\_name=expr[,var\_name=expr]…;

如： delimiter //

CREATE PROCEDURE incre\_num ()

BEGIN

DECLARE a INT ;

DECLARE b INT ;

SET a = 5;

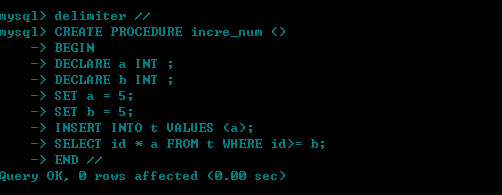
SET b = 5;

INSERT INTO t VALUES (a);

SELECT id \* a FROM t WHERE id>= b;

END //

delimiter ;



MySQL中可以通过select…inot为一个或多个变量赋值

SELECT col\_name [,….] INTO var\_name[,….] table\_expr

2．定义条件和处理程序

定义条件：语法:

DECLARE condition\_name CONDITION FOR [condition\_type]

[condition\_type]：SQLSTATE [VALUE] sqlstate\_value | mysql\_error\_code

定义处理程序：

DECLARE handler\_type HANDLER FOR condition\_value[,..] sp\_statement

handler\_type: CONTINUE | EXIT | UNDO

condition\_value：SQLSTATE [VALUE] sqlstate\_value | condition\_name | SQLWARNGIN | NOT FOUND | SQLEXCEPTION | mysql\_error\_code

3．流程控制

**IF语句：**

IF expr-condition THEN statement\_list

[ELSEIF expr\_conditon THEN statement\_list]…

[ELSE statement\_list]

END IF

如： if val is null

Then select ‘val is null’;

Else select ‘val is not null’’

End if ;

**CASE语句：**

CASE case\_expr

WHEN when\_value THEN statement\_list

[WHEN when\_value Then statement\_list]…

[ELSE statement\_list]

END CASE

如：case val

When 1 then select ‘val is 1’;

When 2 then select ‘val is 2’;

Else select ‘val is not 1 or 2’;

End case ;

**LOOP语句：**

[loop\_label:]LOOP

Statement\_list

End loop [loop\_label]

如：declare id int default 0 ;

Add\_loop:LOOP

Set id=id+1 ;

If id>=10 then leave add\_loop;

End if ;

End loop add\_loop ;

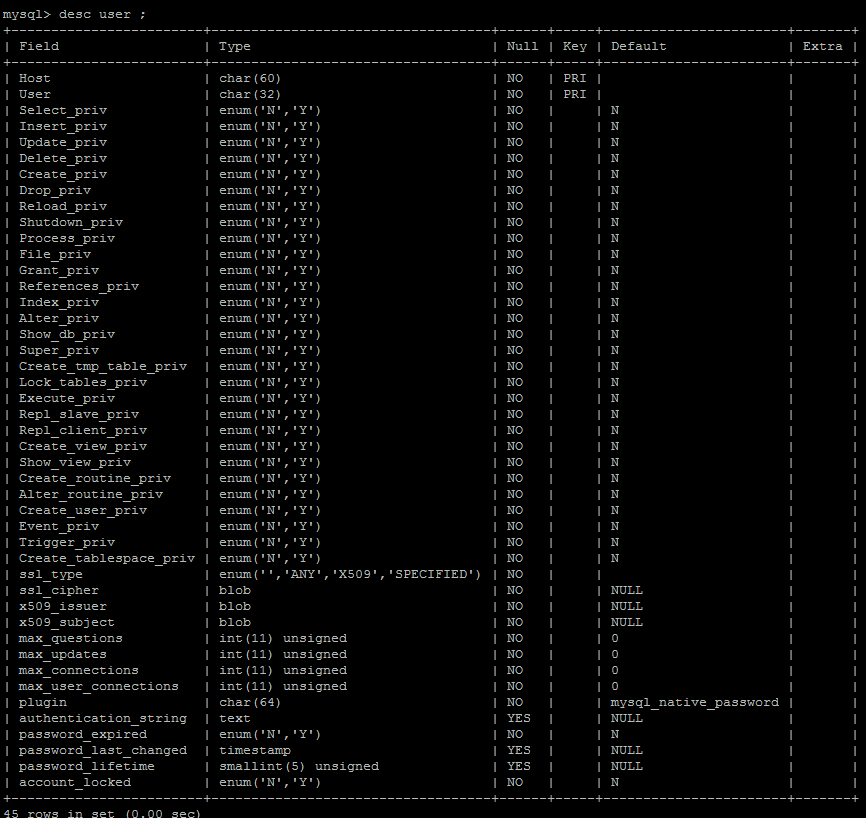
# 用户管理

MySQL是一个多用户数据库，具有功能强大的访问控制系统，可以为不同用户指定允许的权限。MySQL用户可以分为普通用户和root用户。

## 权限表

MySQL服务器是通过权限表来控制用户对数据库的访问，权限表存放在mysq数据库中，有mysqld脚本进行初始化。存储用户权限信息的表主要由：user，db，table\_priv，columns\_priv和procs\_priv。

* user表，允许连接到服务器的账号信息，里面的权限是全局的。



这些字段可以分为四类：用户列，权限列，安全列和资源控制列。

用户列包括：Host，User，authentication\_string，分别表示主机名，用户名和密码。其中User和Host为User的联合主键。

权限列：决定了用户的权限，描述了在全局范围内允许对数据库和数据进行的操作。包括查询权限，修改权限等，还包括关闭服务器，超级权限和加载用户等高级权限。

User表中对应的权限是针对所有用户数据库的。这些权限的字段值的类型为ENUM，可以取的值只能为N或Y，默认都为N，我们可以进行修改。

安全列：只有6个字段，其中两个是ssl相关的，两个是x509相关的，另外两个是授权插件相关的。ssl用于加密；x509标准可用于标识用户；plugin字段表示可以用于验证用户身份的插件，如果该字段为空，服务器使用内建授权验证机制验证用户信息。我们可以使用show variables like ‘have\_openssl’语句查看服务器是否支持ssl功能。

资源控制列：限制用户使用的资源，包含4个字段：

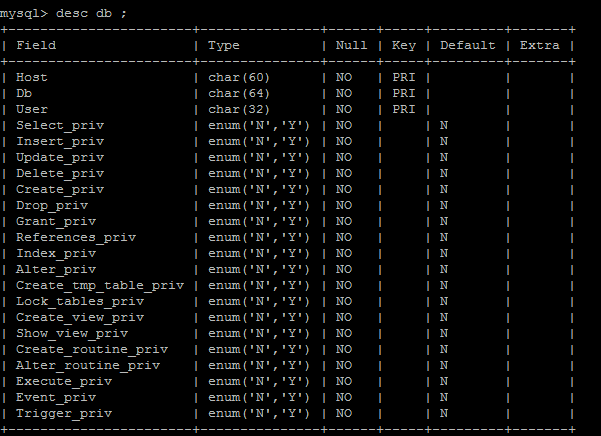
max\_questions：用户每小时允许执行的查询操纵次数。

max\_updates：用户每小时允许执行的更新操纵次数。

max\_connections：用户每小时允许执行的连接操纵数

max\_user\_connections：用户允许同时建立的连接次数。

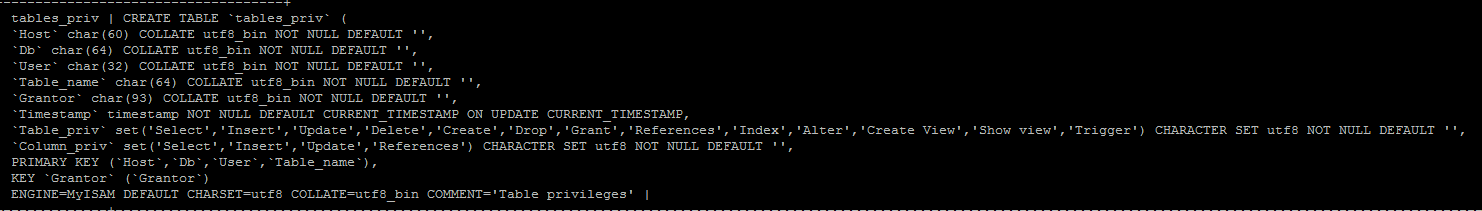
* db表：存储了用户对某个数据操作权限，决定用户能从那个主机存取哪个数据库。host表存储了某个主机对数据库的操作权限，配合db权限表对给定的主机上数据库操作权限做更细致的控制。这个权限表不受GRANy或REVOKE语句的影响。



用户列：有3个字段，分别为Host，Db，User，标识从某个主机连接某个用户对某个数据库的操作权限，这三个字段的组合构建的db表的主键。

* tables\_priv表对表设置操作权限；columns\_priv表对表中的某一列设置权限

tables\_priv表结构：



tables\_priv表有8个字段，分别是Host（主机名），Db（数据库名），User（用户名），Table\_name（表名），Grantor，Timestamp，Table\_priv和Column\_priv。

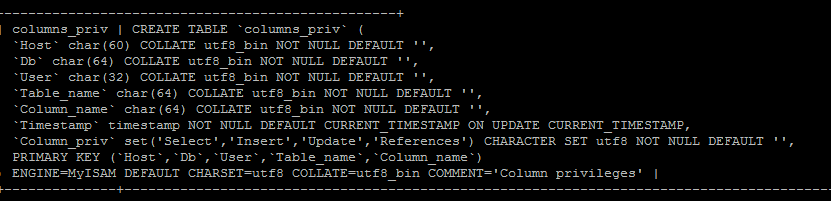
Grantor：表示修改该记录的用户。

Timestamp：表示修改该记录的时间

Table\_priv：对表的操作权限包括：Selector，Insert，Update，Deleter，Create，Drop，Grant，References，Index，Alter，Create View，Show view和Trigger。

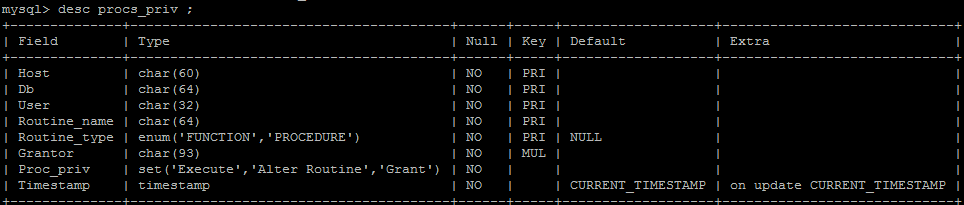
Column\_priv：表示对列的操作权限，包括：Select，Insert，Update，Delete，Creat，References.

columns\_priv表结构：



Column\_name：指定对哪些数据列具有操作权限。

* procs\_priv表可以对存储过程和存储函数设置操作权限。



Routine\_name：存储过程或函数名称。

Routine\_type：存储过程或函数的类型。FUNTION表示这是一个函数，PROCEDURE表示这是一个存储过程。

Grantor：插入或修改该记录的用户。

Timestamp：记录更新时间。

## 账户管理

MySQL提供许多语句用来管理用户管理，这些语句可以用来管理包括登录和退出MySQL服务器，创建用户，删除用户，密码管理和权限管理等。

mysql –help查看mysql命令帮助信息。

### 新建普通用户

1. 使用CREATE USER语句创建新用户

语法格式：

CREATE USER user\_specification[,user\_specification]…….

user\_specification:

user@host

[

IDENTIFIED BY [PASSWORD] ‘password’

IDENTIFIED WITH auth\_plugin [As ‘auth\_string’]

]

user：表示创建用户的名称。

host：表示允许登录的用户主机名

IDENTIFIED BY：表示用来设置用户密码

[PASSWORD]：表示使用哈希值设置密码，该参数可选；

‘password’：表示用户登录时使用的普通明文密码。

IDENTIFIED WITH：为用户指定一个身份验证插件；auth\_plugin是插件的名称，插件的名称可以是一个带有单引号的字符串，或者带引号的字符串。

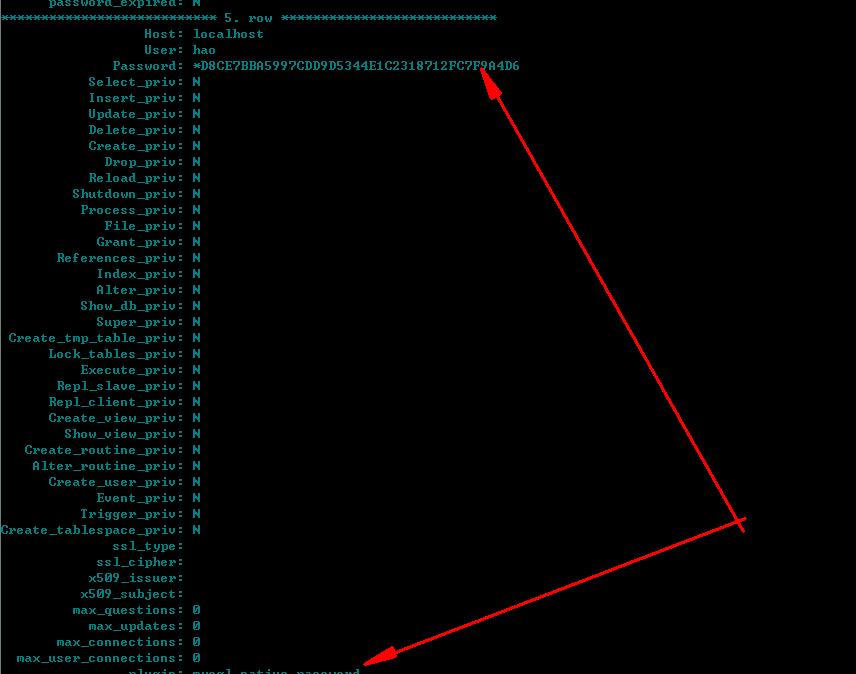
Auth\_string：是可选的字符串参数，该参数将传递给身份验证插件。

**注意：**使用CREATE USER语句，必须要全局的CREATE USER权限或数据库的INSERT权限。

每添加一个用户，CREATE USER语句会在mysql.user表中添加一条新记录，但是新创建的用户没有任何权限。

create user 'hao'@'localhost' identified by 'lsh159';

如果只指定用户名部分，主机名部分则默认为‘%’（即对所有的主机开放权限）



create user 'hao'@'localhost' ;MySQL服务器使用内建的身份验证机制，用户登录时不必指定密码。

create use 'hao'@'localhost' identified by password '\*D8CE7BBA5997CDD9D5344E1C2318712FC7F9A4D6';避免指定明文密码。

Create user ‘hao’@’localhost’ identified with my\_auth\_plugin;服务器调用指定名称的插件，客户端需要提供验证方法所需要的

1. 使用GRANT语句创建新用户

语法格式：

GRANT privileges ON db.table

To user@host [iDENTIFIED By ‘password’] [,user [IDENTIFIED BY ‘password’]]

[WITH GRANT OPTION]

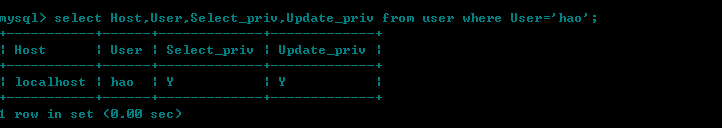
Privileges：表示赋予用户的权限类型。

Db.table：表示用户的权限所作用的数据库中的表

IDENTIFIED BY：用来设置面貌；‘password’：表示用户密码

WITH GRANT OPTION：可选参数，表示对新建立的用户赋予GRANT权限，即该用户可以对其他用户赋予权限。





1. 直接操作MySQL用户表

使用INSERT语句想mysql.user表中插入一条记录来创建新用户。

INSERT INTO mysql.user(Host ,User,Password,[privilegelist]) VALUES(‘’,’’,PASSWORD(‘’),privilegeValueList)；

Privilegelist：表示用户的权限，可以设置多个。

PrivilegeValueList：对应的全向值，只能为’Y’或’N

由于user表中安全列没有默认值，如果想使用该方法，必须自己想办法解决。

### 删除用户

1. DROP USER语句删除用户

语法：DROP USER user [,user]

可以删除多个用户。

如：drop user ‘hao@’localhost;

2．使用delete语句删除。

Delete from mysql.user where Host=’hostname’ and User=’username’;



### 修改密码

1. Root用户修改密码

1.1：使用mysqladmin命令在命令行指定新密码

mysqladmin –u username –p password “new\_password”

1.2：修改mysql.user表

Update mysql.user set Password(‘new\_password’) where User=”username” and Host=”HostName”;

1.3：使用SET语句修改root用户密码

SET PASSWORD=PASSWORD(“NEW\_PASSWORD”);

为了使更改生效，需要重启MySQL或者使用flush privileges;语句刷新权限，重新加载权限表。

1. root用户修改普通用户密码

2.1：使用set语句修改普通用户的密码

SET PASSWORD FOR ‘user’@’host=PASSWORD(“NEW\_PWD)

2.2：使用UPDATE语句修改普通用户密码

UPDATE mysql.user SET PASSWORD=PASSWORD(‘NEW\_PWD) WHERE User=”username” AND Host=”hostname

2.3：使用GRANT语句修改普通用户密码

GRANT USAGE ON \*.\* To ‘username’@’hostname’ IDENTIFIED BY ‘new\_PWD’;

3．普通用户修改密码

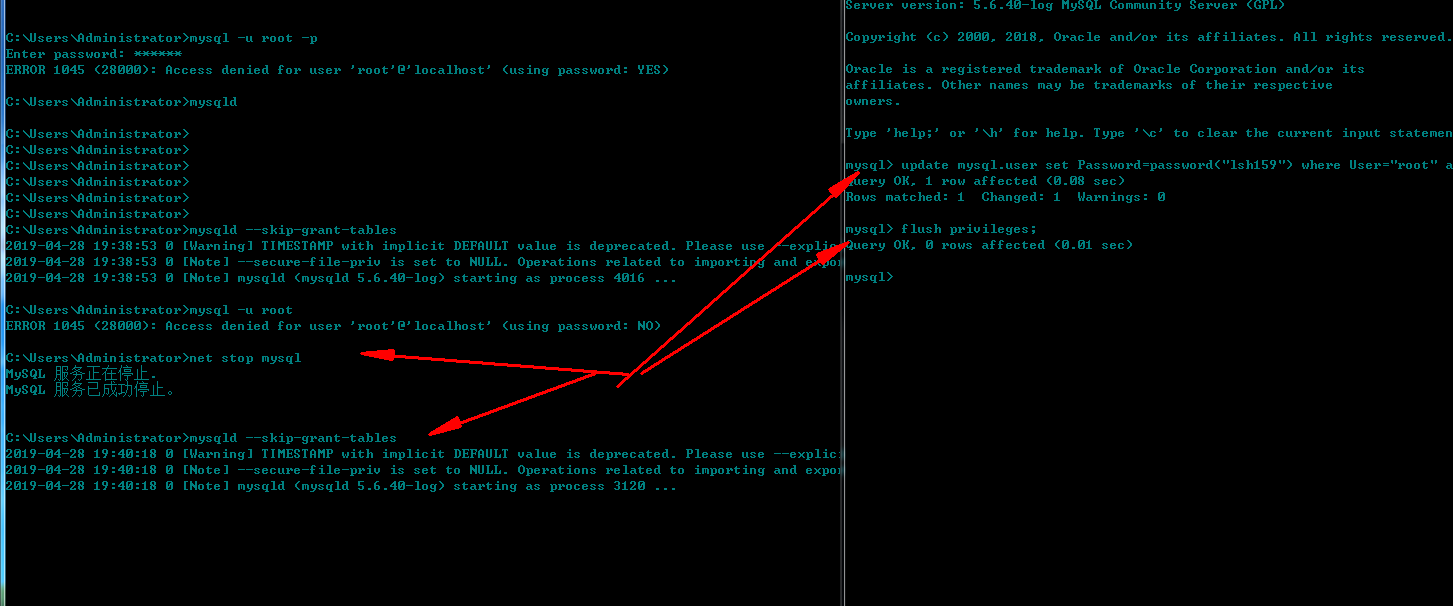
SET PASSWORD=PASSWORD(“NEW\_PWD);

#### 密码找回

1. 使用—skip-grant-tables选项启动MySQL服务器

使用该选线，mysql服务器将不加载权限判断，任何都可以访问数据库。

在window平台，我们使用mysqld或者mysql-nt



## 权限管理

权限管理主要是对登录到MySQL的用户进行权限验证。所有用户的权限都存储在MySQL的权限表中。

GRANT和REVOKE语句中可以使用的权限：

CREATE，DROP，GRANT OPTION，EVENT，ALTER，DELETE，INDEX，INSERT，SELECT，UPDATE，

CREATE TEMPOPARY TABLES，LOCK Tables，TRIGGER，CREATE VIEW，SHOW VIEW，ALTER ROUTINE，

CREATE ROUTINE，EXECUTE，FILE，CREATE TABLESPACE，CREATE USER，PROCESS，RELOAD，REPLICATION CLIENT，REPLICATION SLAVE ，SHOW DATABASES，SHUTDOWN，SUPER.

### 授权

1. 全局层级

适用于一个给定服务器中的所有数据库，这些权限存储于mysql.user表中。

GRANT ALL ON \*.\* TO ‘username’@’hostname’ IDENTIFIED BY password .



REVOKE ALL ON \*.\*

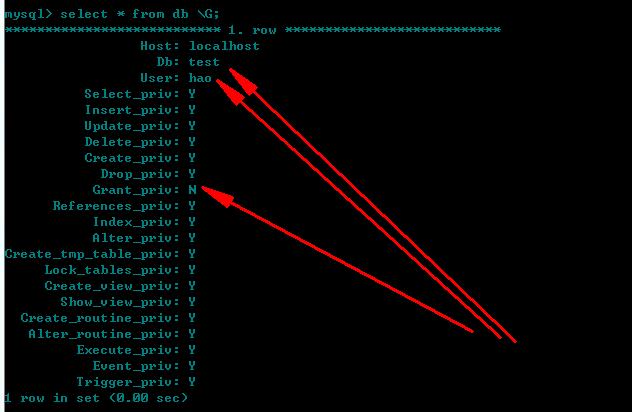


1. 数据库层级

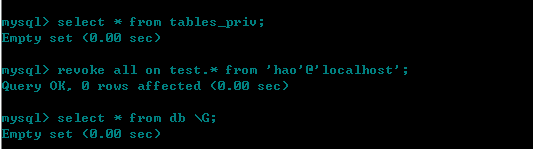
适用于一个给定数据库中的所有目标，存储在mysql.db表中

GRANT ALL ON db\_name.\*





REVOKE ALL ON db\_name.\*

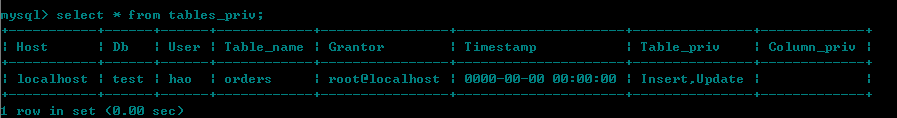


1. 表层级

适用于给定一个表的所有列，存储在mysql.tables.priv表中

GRANT ALLL on db\_name.table\_name To ………………….





REVOKE ALL ON db\_name.table\_name FROM ………………….

1. 列层级

适用于给定表中的每一列，存储在mysql.columns\_priv表中

1. 子程序层级

CREATE ROUTINE ，ALTER ROUTINE，EXECUTE和GRANT权限适用于已存储的子程序，这些权限可以被授予为全局层级和数据库层级。

GRANT语法：

GRANT priv\_type [(columns)] [,priv\_type [(columns)]] ……

ON [object\_type] table1,table2,…tablen

To user [IDENTIFIED BY [PASSWORD] ‘password’] [,IDENTIFIED BY [PASSWORD] ‘password’]

[WITH GRANT OPTION]

Object\_type=TABLE | FUNCTION | PROCEDURE

GRANT OPTION取值：

MAX\_QUERIES\_PER\_HOUR count：设置每小时可以执行count次查询

MAX\_UPDATES\_PER\_HOUR count：设置每小时可以执行count次更新

MAX\_CONNECTIONS\_PER\_HOUR count：设置每小时可以建立count个连接

MAX\_USER\_CONECTIONS count：设置单个用户可以同时建立count个连接。

GRANT OPTION：将自己的权限赋予其他用户。

如：grant select ,insert on \*.\* to ‘hao’@’localhost’ identified by ‘lsh159’ with grant option ;

将全局层级的查询，插入权限赋予hao用户，并可以将权限赋予用户hao创建的用户。

### 回收权限和查看权限

将用户从user表中删除时，应该收回相应用户的所有权限。

1．用户所有权限语法：

REVOKE ALL PRIVILECES GRANT OPTION

FROM ‘username’@’hostname’ [,‘username2’@’hostname2’]

2．回收用户指定的权限：

REVOKE priv\_type [(columns)] [,priv\_type [(coluns)]]….

ON table1 ,table2 ,…,tablen

FROM ‘username’@’password’ [,……]

如：revoke update ,insert on \*.\* from ‘hao’@’localhost’;

Revoke insert delete on db\_ame from ‘hao’@’localhost’;

1. 查看权限

命令：show grants 语句可以显示指定用户的权限信息。

Show grants for ‘username’@’localhost’