# MySql语法大全

**一、 ALTER的 语法**

1.修改表名

**Alter**  **table**  old\_tab\_name **rename**  [**to**]  new\_tab\_name

2.在表的最后一个位置增加字段

**Alter**  **table**  table\_name  **add** 属性名 属性类型

3.在表的第一个位置增加字段

**Alter**  **table**  table\_name  **add**  属性名 属性类型 **first**

4.在表的指定字段后面添加字段

**Alter**  **table**  table\_name  **add**  属性名   属性类型  **after** 属性名

5.删除字段

**Alter**  **table**  table\_name  **drop**  属性名

6.修改字段

**Alter**  **table**  table\_name  **modify**  属性名  数据类型

7.修改字段的名字

**Alter**  **table** table\_name  **change**  旧的属性名 新的属性名 旧数据类型

8.同时修改字段的名字 和 属性

**Alter**  **table**  table\_name  **change**  旧属性名 新属性名  新数据类型

9.修改字段的 顺序

**Alter**  **table** table\_name  **modify**  属性名1  数据类型  **first** | **after** 属性名2

**二、表的完整性约束**

单列约束 ： 即每个约束只是约束一列数据

多列约束 ： 每个约束可以约束多列数据

1. 完整性约束

Not  null   约束字段值不能为空

    Default 设置字段的默认值

    Unique  **key**(UK)     约束的字段值是唯一的

    Primary  **key**(PK)     约束的字段为表的主键，是表的唯一标识

    Auto\_increment     约束字段的值自动添加

    Foreign **key**(FK)     外键

1. 设置唯一约束

**Create**  **table**  table\_name(

属性名  数据类型  **unique**  字段名 ,

);

多字段唯一约束

**Create**  **table**  t\_dept(

deptno  int ,

Dname char(21),

**Constraint**  uk\_name  unqiue(dname)

);

3.多字段主键

**Create**  **table**   table\_name(

属性名  数据类型，

         【 **constraint** 约束名 】primary  **key** (属性名 ， 属性名 .... )

);

1. 外键约束 【 设置了FK 约束必须依赖于数据库中以及存在的父表的 主键 】

**Create**  **table** table\_name (

属性名  数据类型，

...

**Constraint**  外键约束名  foreign  **key**  (属性名1)

**References**  父表名（字段名1）

)

**三、索引的操作（mysql 数据库支持至少 16 个索引）**

INNODB  默认使用的存储引擎支持 **BTREE**类型索引

MerMory 存储引擎支持HASH类型索引

1.索引的分类（6种）

普通索引

唯一索引

全文索引

单列索引

多列索引

空间索引

2.何时使用索引

经常被查询的字段， 即在where 子句中出现的字段

在分组中的字段， 即在 **group** **by** 下的字段

存在依赖关系的字段，如主键和外键

拥有许多重复值的字段

3.普通索引的创建

创 建表的时候 创建索引

**Create** **table** tab\_name(

字段名  数据类型，

...

**Index** | **key** 【索引名】 （属性名1 【长度】 【 **asc** | **desc** 】）

)

在建好的表上创建索引

**Create** **index**  索引名  **on** 表名（属性名 【长度】 【 **asc** | **desc** 】）

使用**alter** **table** 创建索引

**Alter**  **table**  tab\_name  **add**  **index** | **key**  索引名（属性名 【长度】 【 **asc** | **desc** 】）

4.唯一索引的创建

创 建表的时候 创建索引

**Create** **table** tab\_name(

字段名  数据类型，

...

Unqiue  **Index** | **key** 【索引名】 （属性名1 【长度】 【 **asc** | **desc** 】）

)

在建好的表上创建索引

**Create**  **unique**   **index**  索引名  **on** 表名（属性名 【长度】 【 **asc** | **desc** 】）

使用**alter** **table** 创建索引

**Alter**  **table**  tab\_name  **add**  **unique**  **index** | **key**  索引名（属性名 【长度】 【 **asc** | **desc** 】）

5.全文索引的 创建

创 建表的时候 创建索引

**Create** **table** tab\_name(

字段名  数据类型，

...

Fulltext  **Index** | **key** 【索引名】 （属性名1 【长度】 【 **asc** | **desc** 】）

)

在建好的表上创建索引

**Create** fulltext  **index**  索引名  **on** 表名（属性名 【长度】 【 **asc** | **desc** 】）

使用**alter** **table** 创建索引

**Alter**  **table**  tab\_name  **add**  fulltext  **index** | **key**  索引名（属性名 【长度】 【 **asc** | **desc** 】）

6.多列索引的创建

创 建表的时候 创建索引

**Create** **table** tab\_name(

字段名  数据类型，

...

**Index** | **key** 【索引名】 （属性名1 【长度】 【 **asc** | **desc** 】,

....,

属性名**n** 【长度】 【**asc** | **desc** 】）

)

在建好的表上创建索引

**Create** **index**  索引名  **on** 表名（属性名 【长度】 【 **asc** | **desc** 】,

....

属性名**n**【长度】 【**asc** | **desc** 】

）

使用**alter** **table** 创建索引

**Alter**  **table**  tab\_name  **add**  **index** | **key**  索引名（属性名 【长度】 【 **asc** | **desc** 】,

....

属性名**n** 【长度】 【**asc** | **desc** 】

）

7.删除索引

**Drop** **index**  index\_name  **on**  table\_name

8.查看索引

**Explain**  **select**  \*  **from**  table\_name  **where** .....

**四、视图的操作**

* 对于视图数据可以进行增删改 操作
* 视图来自多个表的 时候不允许增删数据

1.视图的 创建

**Create**  **view**  view\_name  **as**  查询语句

2.删除视图

**Drop**  **view**  view\_name ;

3.查询视图

**Select**  查询字段  **from**  view\_name

4.查看视图的 相关信息

**Show** **table** **status**  **like**  ‘**view**’ ;

**Show** **table** **status**  **like**  ‘view\_name(自定义的**view** )’ ;

5.Show create view view\_name

6.通过系统表（information\_schema） 查看视图信息

**Use**  information\_schema ;

**Select**  \*  **from**  views  **where**  table\_name  =’view\_name(自定义的视图名)’;

7.修改视图

替换或是创建视图的方法

**Create**  **or**  **replace**  **view**  view\_name  **as**  查询语句

**Alter** 语句修改视图

**Alter**  **view**  view\_name  **as**  查询语句

**五、触发器的一操作**

1.在触发器进行下面操作语句时， 会触发触发器

**Delete**   **insert**   **update**

2.创建单条语句的 触发器 trigger

**Create**  **trigger**  trigger\_name

**Before** | **after**  触发事件(**update** | **insert** | **delete**)

**On** 表名 **for**  **each**  **row**

触发的语句

3.创建多条触发器

Creater  trigger  trigger\_name

Befor | after  触发事件(**update** | **insert** | **delete**)

**On**  表名  **for**  **each**  **row**

**Begin**

触发的语句

**End**

4.查触发器

**Show** **triggers** ;

5.通过系统表来 查看触发器

**Use** information\_schema

**Select**  \*  **from**  **triggers**    //查看所有的触发器

**Select**  \*  **from**  trigger\_name(查询的触发器名)

6.删除触发器

Drop  trigger   trigger\_name

**六、单表查询数据记录**

主要学习内容有：

    简单数据记录查询

    条件数据记录查询

    排序数据记录查询

    限制数据记录查询

    统计函数 和 分组数据记录查询

1.避免 重复数据查询

**Select**  **distinct**  field1 ,field2  **from**  table\_name ;

2.设置格式显示数据查询

**Select**  **CONCAT**(filed1 , ‘文本解释内容’, **value** ) **from** table\_name;

**Select**  **concat**(ename,’雇员年薪是：’,sal\*12 ) **from** emp;

1. 条件数据 记录查询

带有关系运算符和 逻辑运算符的 数据查询

带有**between** **and** 关键字的 条件查询

带有**is** null 关键字的 条件记录查询

带有 **in** 的条件记录查询

带有  like 的条件记录查询

A.between  and

**Select** filed1 , field2 ...  **From** table\_name  **where**  **field**  **between** value1 **and**  value2

B. 不符合 范围的数据记录查询

**Select** filed ...  **From** table\_name  **where** filed  **not**  **between** value1 **and** value2

**C**. 带 **IS** <**NOT**>  NULL 的关键字查询

**Select**  **field** ....  table\_name **where**  filed  **is** <**not**>  null

**D**. 带 <**not**> **in**  关键字 的 集合查询

**Select** filed ... **From** table\_name  **where**  filed <**not**> **in** (value1,value2,value3,....)

    该情况类似于：

**Select**  filed **form** table\_name  **where**  filed=value1 **or**  filed=value2 **or** ....

**E**.带**like** 的模糊查询

**Select** filed ....  **From**  table\_name  **where**  filed  **like** **value**;

**Select** filed .,.. **From**  table\_name   **where**   filed   **not**  **like** **value**

**Select** filed .,.. **From**  table\_name   **where**   **not**   filed   **like** **value**

    <

    “\_”    该字符表示， 通配符能匹配单个字符

    “%”   该通配符表示可匹配任意字符的模糊查询

    >

4.排序数据记录查询

**Select**   filed  ...  **From**  table\_name  **where**   条件(contition  )  **order**  **by**  filed1\_name   [**asc** | **desc** ] ,  file2\_name [ **asc** | **desc** ]

5.限制 数据记录查询 数量

**Select**  filed ....  **From**  table\_name  **where**   条件语句 **limit**  offser\_start , row\_counts;

6.统计记录查询 (avg count sum max min )

**Select**  **function**(filed)  from  table\_name  **where**  条件语句 ;

\*关于统计函数的 注意点

对于mysql 的统计函数 ，如果操作的表中没有任何数据记录 ， 则 count() 函数会返回数据0 ， 其他的 函数 返回值为  null

7.分组数据 查询

**Select**  **function**()  **from**   table\_name  **where**  条件语句  **group**  **by**  filed;

8.实现统计功能 的 分类查询

**Select**   **group\_concat**(filed)  **from**  table\_name  **where**  条件语句  **group**  **by**  filed;

10.实现多个分组的 查询

**Select**  **group\_concat**(filed), **function**(filed)  **From**  table\_name  **where**  condition  **group**  **by**   Filed1 , filed2  ...;

11.分组的 数据查询 -- 实现 having 子句限定分组查询

**Select**  **function**(filed) **from** table\_name  **where**  **group**  **by**  filed1 ,filed2

**Having**  contition

**七、多表数据记录查询**

1.内连接查询

**Select** filed1 ,filed2 ..... Filedn   **from**  join\_table\_name1  **inner**  **join**  join\_table\_name2  [ **inner** **join**  join\_table\_name\_n ]  **on**   连接条件

例子：

**Select**  **e**.empno  , **e**.ename , **e**.sal , **e**.job , **l**.ename  **from**  t\_employee  **e**  **inner**  **join**  t\_employee  **l**  **on**   **e**.agr =  **l**.empno ;

**select**  **e**.empno  ,  **e**.ename  , **e**.sal  ,**e**.job , **l**.ename ,**d**.dname  ,**d**.location  **from** t\_employee  , t\_employee  **l** ,  t\_dept  **d**  **where**   **e**.mgr =**l**.empno a **and**  **l**.deptno  = **d**.deptno  ;

2.外连接查询

**select**  filed1,  filed2  ... Filed\_n   **from**   join\_table\_name1

**Left** | **right** | **full**  **join**  join\_table\_name2  **on**  连接条件

3.合并查询数据记录

select  filed1 , filed2 .....  Filedn  from  tablename1

Union  | **union**  **all**

Select  filed1,  filed2  ....  Filedn  from  tablename2

.....

4.子查询

带 in 的子查询

**Select**  \*  **from**  t\_employee  **where**  deptno  **in**  (**select**  deptno  **from**  t\_dept)

带 **any**  的子查询

=**any** ： 它的功能与关键字一样

>**any**(>=**any**) ： 比子查询中返回的数据记录最小的还要大于数据记录

**Select**  ename ,  sal  **from**  r\_employee  **where**  sal  > **any**  ( **select**  sal  **from** t\_employee  **where**  job=’manager’ )

返回结果为多列的列子查询

**Select**  **count**(\*)  number  **from**  t\_dept  **d** ,

( **select**  deptno dno , **count**(empno)  number , **avg**(sal)  average  **from**  t\_employee  **group**  **by**  deptno )   employee ;

**八、使用mysql 常用函数**

字符串函数

数值函数

日期函数

系统信息函数

1.常用的字符函数

1)合并字符串的函数 concat()  和 concat\_ws() 函数

    Concat ( s1 ,s2c , ... S\_n )

    Concat\_ws( sep ,  s1 , s2 ... S\_n );  -- sep 为分隔符

    例子： Concat\_ws( ‘-’ , ‘098’ , ’75914’  );

2) 比较字符串大小函数 **strcmp()**

**Strcmp**( **str1**, **str2**);

3) 获取字符串长度length()  和 char\_length()

    Length( **str**)

    Char\_length( **str**)

4) 实现大小写转换函数 upper()  和 字符数函数  lower()

    Upper()  函数 等同于 ucase()

    Lower()  函数等同于  lcase()

5) 查找字符串位置函数

    Find\_in\_set( **str1, str2**)            / /返回字符串函数位置

    Filed( **str** ,**str1**,  **str2**, **str\_n**.... )   / /返回字符串指

    定位置的函数

    Locate( **str1, str2**)

    Position ( **str1** in  **str2**)

    Instr ( **str**, **str1**)                 // 查找字符串匹配的位置

6) 截取字符串函数

    Left ( **str**,num )

    Right( **str** ,  num )

**Substring( str**,num , len  )   //截取指定位置和长度的字符串

7)替换字符串  insert() 函数  replace () 函数

    Insert( **str**, pos , length , 替换后的字符 ) ;

    Replace ( **str**, 替换的内容 , 替换后的字符);

8)去除首尾空格函数

    Ltrim( **str**)

    Rtrim( **str**)

    Trim( **str**)

2.使用数值函数

Abs (x)  返回数值 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)           返回x 模 y 的值

**Rand**(x) 返回 0 ~ 1 之间的随机数

**Round**(x, y) 返回数值 x 的四舍五入后有 y 位 小数的数值

**Truncate**(x ,y) 返回数值 x 截取 位数为 y 位的数值

3、使用日期函数和时间函数

Curdate() 和 current\_date() 获取的前日期

Curtime() 和 current\_time() 获取当前时间

Now()                                 获取当前的日期时间

Unix\_timestamp(date) 获取当前date 的 unix 时间戳

Form\_unixtime() 获取 unix 时间戳的日期值

Utc\_date() 国际协调时间

Utc\_time()

Week(date)   返回当前日期是一年中的那一天

Year(date)   返回日期date的年份

Hour(time) 返回时间的小时值

Minute(time) 分钟值

Second(time) 秒数值

Monthname(date) 返回时间的月份值

例子：

Select  now()  当前时间和日期,

YEAR( NOW()) 年,

-- Quarter(now()) 季度 ,

Month(now()) 月份,

Week(now())    星期,

Dayofmonth( now()) 天,

Hour(now()) 小时 ;

1)  获取指定值的函数 extract( )

    Select  Extract( type  from  date )

    例子：select  extract( year from  now() );

2) 关于星期的 函数

    Dayname() 星期几的英文名

    Dayofweek()     返回是一星期中的第几天

    Weekday() 返回星期几

3) 计算日期和 时间的函数

    To\_days(date)

    From\_days(date())

    Datediff( date1 , date2)

    例子：

    Select  now() , to\_days(now()) 相隔天数,

    From\_days(to\_days(now()) 一段时间后的日期和时间,

    Datediff(now() , ‘2000-1-1’);

    Select datediff(now() , ‘2000-1-1’);

4)  于指定日期时间的操作

    Adddate(date , n) 表示 n 天后的日期

    Sundate(date ,n) 表示 n 天前的日期

    Addtime(time , n) 表示 n 秒后的时间

    Subtime(time , n) 表示 n 秒前的时间

4.使用系统信息函数

Version() 返回数据库版本号

Datebase() 返回当前数据库名字

User() 返回当前用户

Last\_insert\_id()   返回最近生成的 auto\_increment 值

例子：

Select  version() , user() ,datebase();

5.实现特殊功能的函数

**Password**(str) 对于字符串 str 进行加密

**Format**(x , n) 实现将数字 x 进行格式化 ，保留n 位小数

**Inet\_aton**(ip) 把 ip 地址 转换成数字

**Inrt\_ntoa**(x) 实现将数字转换成 ip

Get\_loct (name , time) 新建一个持续时间 为 time 的名为 name 的锁

**Release\_loct**(name) 解锁

**Benchmark**( counter  , expr) 把表达式执行 count 次

**Convert**(s using cs) 实现将字符串s 的字符集变成 cs

**Convert**(x ,type) 实现将x 变成 type 类型

**九、mysql 的存储过程和 函数的操作**

1.创建存储过程

**Create**  **procedure**  **procedure\_name**  **begin**   ---   **end**  ;

2.创建函数

Create  **function**  **function**\_name

3.定义存储过程 和 函数 的参数

Parameter\_name   **type**   ;

如：  **name**  int ;

4.关于存储过程 和函数的表达式

1） 声明变量

**Declare**  var\_name  **type**  【 **default**  **value** 】

2) 赋值变量

**Set**  var\_name = expr

3) 使用 **select** .... **Into** 实现赋值

**Select**  filed\_name  **into**  var\_name   **from**   table\_name  **where**  condition（条件）;

4) 定义条件

**Declare**  condition\_name   condition  **for**  condition\_value  ;

5.删除存储过程 和 函数

Drop  **procedure**  **procedure\_name**  ;

Drop  **function**   **function\_name** ;

1. 修改 存储过程 和函数  
   Alter procedure procedure\_name [ 修改 内容语句 ]  
   Alter function function\_name [ 修改 语句 ]

7.使用流程语句

**If**  search\_condition  **then**  statement\_list

. . .

**End** **if**

**Case**  case\_value

**When**  when\_value  **then**   statement\_list

...

**Else**  statement\_list

**End**  **case**

8.循环控制语句

1) [ begin\_label : ]  **loop**

Statement\_list

**End**  **loop**  [ end\_loop ]

Leave  begin\_label  // 离开标签

2)   [ begin\_label : ]  **while**  searcg\_condition **do**

Statement\_list

**End** **while** [ **end** lable ]

3)   [ begin\_label : ]  **repeat**  search\_condition  **do**

Statement\_list

**End**  **repeat**  [ end\_list ]

9.查看 存储过程 和 函数

1）通过 **show** **status**  语句查看函数和存储过程

**Show** **procedure** **status**  [ **like**  ‘pattern’ ]

**Show** **function**  stutus  [ **like**  ‘pattern’ ]

2）通过 系统表 information\_schema.routines  查看函数和存储过程

**Use**  information\_schema

**Select**  \*  **from**  routines  //

3）通过 **show**  **create**  语句查看函数和存储过程]

**十、mysql 游标的 使用**

1.声明 光标

Decade  cursor\_name  cursor  from  select\_statement[sleect查询语句];

2.打开光标

**Open**   cursor\_name

3.使用 光标

Fetch  cursor\_name **into**   var\_name ....

4.关闭光标

Close  cursor\_name;

**十一、mysql 的日志管理**

1.二进制日志： 该日志文件会以二进制的形式记录下数据库的各种操作，但是不会记录查询语句

1）修改my.ini：

[mysqld]

Log-bin[=save\_bin\_path\filename]   //保存二进制文件的目录

2）查看二进制文件

Mysqlbinlog  filename.number

3)停止或是 开启二进制文件

**Set** sql\_log=0 ;   //关闭

**Set** sql\_log=1 ; //开启

4）删除二进制文件

**Reset** mater  ; -- 删除索引的二进制文件

**Purge** **master** **logs**  **to**  filename.number ; -- 删除所有编号小于二 进制文件的日志文件

**Purge**  **master**  **logs**  **before**  ‘yyyy-mm-dd’ ; --删除指定时间的日 志文件

5) 利用 mysql 二进制文件进行恢复数据库

-- 恢复在时间点前开始的数据备份

Mysqlbinlog  -- start  - datetime = ‘ 时间点’  -u root  -p

-- 恢复在时间点后的数据备份

Mysqlbinlog  --stop  -datetime =’时间点’  -u  root  -p

-- 恢复在 某一个位置 的 数据备份

Mysqlbinlog --start【stop】  -position=’该位置的id’  -u  root  -p

2.错误日志： 该日志文件 会记录mysql 的启动 、关闭 、运行时出错的信息

1）修改 my.ini 文件

[mysqld]

Error-bin[=path\filename]

2）删除错误日志文件

Mysqladmin  -u  root  -p  flush  -logs

3.其他类型日志

查询日志  ：该类型 的日志包括  通用查询日志 和 慢查询日志

通用查询日志： 记录用户登录和记录查询语句

慢查询 日志： 记录执行时间超过 指定时间的 各种操作

1)修改 my.ini

[mysqld]

Log [=path/filename]

2)配置慢查询文件

[mysqld]

Log-slow-queries [=path\filename]

Long\_query\_time=n -- 设定时间

3）删除慢查询日志文件

Mysqladmin -u root  -p  flush  -logs

**十二、mysql 的维护和 性能提高**

1.通过 mysqldump 是显示数据备份

1) 备份一个数据库

Mysqldump  -u username  -p  dbname  table\_name  > backup\_name.sql

2）备份多个数据库

Mysqldump -u username  -p  --datebases  dbname  , dbnamen .... > path\database\_back.sql

3）备份所有数据库

Mysqldump  -u  username  -p  --all-datebases  > backup\_name.sql

2.使用 mysqldump 还原数据库

Mysqldump  -u  username  -p  [ dbname ] <  backup\_name.sql

3.实现数据库中的表导出成文本文件

1）利用 **select** ...  **Into**  **outfile** 方式来实现导出操作

**Select**  [ filed\_name ]  **from**  table\_name  [ **where** contion ]  **into**  **outfile**  ‘filename’  [ **option** ]

例子：

**Select**  \*  **from**  t\_dept  **into**  **outfile**   ‘**c**:/file\_name.txt ’  --输出的文件

Fileds  **terminated** **by**  ‘\,’   -- 段结束符

**Optionally**  **enclosed**  **by**  ‘\ “’ --字段结束符

**Lines**  **starting**  **by**  ‘\>’ --行结束符

**Terminated**  **by**  ‘\r\**n**’ ;

2）利用mysqldump 命令实现 导出的操作

Mysqldump  -u  root  -p  -T  c:\ database\_name.txt

3）利用mysql 命令来实现导出操作

Mysql  -uroot  -p  -e  “**select**  [ file\_name ]  **from**  table\_name”  dbname > file\_name;

4.实现文本文件导入到数据库表

1）使用 “**load**  **data**  **infile**” 方式导入数据

**Load**  **data** [**local** ]  **infile**  file\_name  **into**  **table**  table\_name  [ **option** ]

2）使用 mysqlimport  命令来实现导入操作

Mysqlimport  -u root  -**p**  [ --local ]  dbname  file\_name  [ option ]

**十三、mysql 的安全机制**

1.创建普通用户

1）使用 **create** **user** 创建用户

**Create**  **user**  username  [ **identified**  **by**  **password** ]

2）使用**insert** 语句创建用户

**Insert** **into**  **user**( host , **user** , **password** )  **values**( ‘’ , ’’ , ’’);

执行 一下语句使其生效

**Flush**  **privileges** ;

3) 执行 **grant** 语句来创建用户

**Grant**  priv\_type **on**  databasesname.**table**  **to**

  username  [ **identified**  **by**  **password**  ]

...

2.修改用户密码

1）使用 mysqladmin 修改用户密码

 Mysqladmin -u  username  -p  password  new\_password

2） 用户登录后 可以修改 用户的密码

**Set**   **password** = **password**( ‘new\_password’);

3) 更新系统表 mysql.user 数据记录修改密码( 限于 root 用户 )

**Update** **user**  **set**  **password**=**password**(‘new\_password’)

**Where**  **user** =”user\_name” **and**  host = ‘localhost’ ;

4）使用**grant** 来修改密码

**Grant**  priv\_type **on**  databasesname.**table**  **to**

  username  [ **identified**  **by**  “new\_password”  ]

3.删除 普通用户账号

1) 使用 drop **user** **语句**

**Drop** **user**  **user1**  ,  user2....

2）使用 mysql.**user** **表来实现删除 普通用户账号**

**Delete**  from  **user**  **where**= ‘user\_name’  and  host = ‘ localhost ’ ;

4.权限管理

1）对用户进行授权

**Grant** priv\_type  [ (column\_list) ] **on**  **database**.**table**  **to**  **user**  [ **identified**  **by**  **password**  ]  **with**  with\_option

2) 授权的权限 有

**Grant**  **option** ： 被授权用户可以授权给其他用户

Max\_queries\_per\_hour **count**： 每小时的最大查询数

Max\_connections\_per\_hour **count**： 每小时的最大连接数

Max\_updates\_per\_hour **count**： 每小时的最大更新数

Max\_user\_connection **count**： 单个用户可以同时有**count**个连接

3) 查看用户权限

A)查看指定用户权限

**Select**  host  , **user** , **password** , select\_priv , update\_priv , grant\_priv , drop\_priv  **from**  mysql.**user**  **where**  **user**=’user\_name’

B)查看所有用户的权限

**Show**  **grants**  **from**  ‘user\_name’ ;

5.收回权限

1) 收回所有权限

**Revoke**  all  **privileges** , **grant**  **option**  **from**  user\_name [ **identified**  **by**  **password** ]

2) 收回权限

**Revoke**  priv\_type  [( column\_list )]  **on**  **database**.table\_name  **from**  user\_name  [ **identified**  **by**  **password**  ]