

MySQL 高级教程.....	4
1、概述	4
一、 视图	5
1. 视图的定义	5
2. 视图的作用	5
(1) 可以简化查询:.....	5
(2) 可以进行权限控制，	6
3. 查询视图	8
4. 修改视图	8
5. 删除视图	8
6. 查看视图结构	8
7. 查看所有视图	8
8. 视图与表的关系	8
9. 视图算法	9
二、SQL 编程.....	9
1、变量声明	9
(1) 会话变量	9
(2) 普通变量	9
(3) 变量赋值形式	10
2、运算符	11
(1) 算术运算符	11
(2) 关系运算符	11
(3) 逻辑运算符	11
3、语句块包含符	11
4、If 判断.....	12
5、case 判断.....	14
6、循环.....	15
(1) loop 循环.....	15
(2) while 循环	16
(3) repeat 循环	16

三、存储过程.....	17
1、 概念	17
2、 存储过程的优点	17
3、 创建存储过程	17
4、 调用存储过程	18
5、 删除存储过程	18
6、 创建复杂的存储过程	18
7、 declare 声明局部变量	19
8、 用户变量	19
9、 系统变量	19
四、函数.....	20
1、自定义函数	20
(1) 定义语法:	20
(2) 调用	20
2、系统函数	22
(1) 数字类	22
(2) 大小写转换	22
(3) 截取字符串	22
(4) 时间类	22
五、触发器.....	26
1、简介	26
2、触发器四要素	26
3、创建触发器	27
4、删除触发器	33
5、查看触发器	33
6、before 与 after 的区别	33
六、游标.....	34
1. 创建游标	34
2. 打开游标	35
3. 使用游标	35

4.	关闭游标	35
5.	游标实例	35
七、MySQL 查询.....		36
1, 从而实现排序;.....		36
2、sql1{不管数据相同与否, 排名依次排序(1,2,3,4,5,6,7...)}.....		38
2、sql2{只要数据有相同的排名就一样, 排名一次排序(1,2,2,3,3,4,5....)}.....		39
3、sql2{只要数据有相同的排名就一样, 但是相同排名也占位, 排名一次排序 (1,2,2,4,5,5,7....)}		40
4、基本函数		42
阶乘函数		42
递增数列		42
随机数字		42
MySQL 高效编程—学习笔记		43
一、MySQL 基础篇.....		43
二、MySQL 高级应用篇.....		48
三、Shell 高级编程企业实战题以及参考答案		59
1、监控 MySQL 主从同步		59
2、批量创建文件以及改名		65
3、批量创建用户随机密码		70
4、判断网络主机存活		72
5、解决 DOS 攻击生产案例		73
6、MySQL 启动脚本.....		75
7、分库备份		80
8、分库分表备份		81
9、打印字母数不大于 6 的单词		82
10、比较 2 个整数大小		89
11、打印选择菜单		89
12、监控 web、db 服务		89
13、监控 memcache 服务		89
14、监控 web 站点目录		89

15、rsync 的系统启动脚本.....	89
16、抽奖.....	89
17、破解密码.....	89
18、批量检查多个网站地址是否正常.....	89
MySQL---Open 实例.....	90
1、MySQL 查询今天、昨天、近 7 天，近 30 天，本月，上一月的数据.....	90
2、统计时间分布脚本.....	90
2.1、查询超时记录.....	90
2.2、导出日期范围内的日志信息(已 xml 格式导出为 stat.xml,使用 navicat 工具导入进 stat 表).....	91
2.3、统计超时率(mysql 库中).....	91
2.4、时间分布(异常统计).....	91
2.5、日期分布(正常统计).....	93
3、连接 MySQL 数据库 PHP 代码.....	94
4、Ruby 连接 mysql 代码.....	94
5、mysql 批量删除.....	95
6、MySQL 中取得汉字字段的各汉字首字母.....	95
7、MySQL 计算日期之间相差的天数.....	97
9、MySQL 存储过程代码例子.....	97
10、php 连接 mysql 的工具类.....	98
11、MySQL 经纬度进行距离计算.....	108
12、MySQL 时间函数.....	109
13、MySQL 游标使用模板.....	109
14、MySQL 任务调度实现.....	110
15、php 调用 mysql 存储过程返回结果集.....	113

MySQL 高级教程

零、概述

类型	存储	最小值	最大值
----	----	-----	-----

	(Bytes)	(Signed/Unsigned)	(Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

INTEGER 同 INT

一、视图

1. 视图的定义

视图的定义: 视图是由查询结果形成的一张虚拟表, 是表通过某种运算得到的一个投影。同一张表可以创建多个视图

创建视图的语法:

Create view view_name as select 语句

说明:

- 视图名跟表名是一个级别的名字, 隶属于数据库;
- 该语句的含义可以理解为: 就是将该 select 命名为该名字(视图名);
- 视图也可以设定自己的字段名, 而不是 select 语句本身的字段名--通常不设置。
- 视图的使用, 几乎跟表

2. 视图的作用

(1) 可以简化查询:

案例 1: 查询平均价格前 3 高的栏目。

传统的 SQL 语句:

```
Select  cat_id, avg(shop_price) as pj from esc_goods group by cat_id order by pj desc
limit 3;
```

```
mysql> select cat_id,avg(shop_price) pj from ecs_goods group by cat_id order by pj desc;
+-----+-----+
| cat_id | pj      |
+-----+-----+
| 5      | 3700.000000 |
| 4      | 2297.000000 |
| 3      | 1746.066667 |
| 2      | 823.330000  |
| 8      | 75.333333   |
| 15     | 70.000000   |
| 14     | 54.000000   |
| 13     | 33.500000   |
| 11     | 31.000000   |
+-----+-----+
9 rows in set (0.04 sec)
```

```
mysql> select cat_id,avg(shop_price) pj from ecs_goods group by cat_id order by pj desc limit 3;
+-----+-----+
| cat_id | pj      |
+-----+-----+
| 5      | 3700.000000 |
| 4      | 2297.000000 |
| 3      | 1746.066667 |
+-----+-----+
3 rows in set (0.00 sec)
```

创建一个视图，简化查询。

语法：

Create view ecs_goods_v1 as select cat_id, avg(shop_price) as pj from ecs_goods group by cat_id

Select * from ecs_goods_v1 order by pj desc limit 3;

```
mysql> select * from ecs_goods_v1 order by pj desc limit 3;
+-----+-----+
| cat_id | pj      |
+-----+-----+
| 5      | 3700.000000 |
| 4      | 2297.000000 |
| 3      | 1746.066667 |
+-----+-----+
3 rows in set (0.00 sec)
```

案例 2: 查询出商品表，以及所在的栏目名称；

传统写法：

Select goods_id, goods_name, b. cat_name, shop_price from ecs_goods a left join ecs_category b on a. cat_id = b. cat_id;

创建一个视图

Create view ecs_goods_v2 as select goods_id, goods_name, b. cat_name, shop_price from ecs_goods a left join ecs_category b on a. cat_id = b. cat_id;

查询视图：

Select * from ecs_goods_v2;

(2) 可以进行权限控制，

把表的权限封闭，但是开放相应的视图权限，视图里只开放部分数据，比如某张表，用户表为例，2 个网站搞合作，可以查询对方网站的用户，需要想对方开放用户表的权限，但是呢，又不想开放用户表的密码字段。

再比如一个 goods 表，两个网站搞合作，可以相互查询对方的商品表，比如进货价格字段不能让对方查看。

```
mysql> select * from goods;
```

id	goods_name	in_price	shop_price
1	dog	10.00	10000.00
2	pig	50.00	18000.00
3	monkey	150.00	150000.00
4	cat	0.00	8000.00

4 rows in set (0.00 sec)

第一步:创建一个视图，视图中不能包含 in_price 字段。

语法: create view goods_v1 as select id,goods_name,shop_price from goods;

第二步: 创建一个用户，授予查询权限，只能操作 goods_v1 表（视图）

语法: grant select on php.goods_v1 to 'xiaotian'@'% 'identified by '123456';

第三部:我们就测试以下，

```
<?php
```

```
$conn = mysql_connect('localhost','xiaotian','123456');
```

```
Mysql_query('user php');
```

```
Mysql__query('set names utf8');
```

```
$sql = "select * from goods_v1";
```

```
$res = mysql_query($sql);
```

```
$data = array();
```

```
While($row = mysql_fetch_assoc($res)) {
```

```
    $data[] = $row;
```

```
}
```

```
Echo '<pre>';
```

```
Print_r ($data);
```

```
?>
```

3. 查询视图

语法: `select *from 视图名 [where 条件]`

视图和表一样，可以添加 `where` 条件

4. 修改视图

`Alter view view_name as select 语句`

5. 删除视图

`Drop view 视图名称;`

6. 查看视图结构

和表一样的，语法，`desc 视图名称`

7. 查看所有视图

和表一样，语法：`show tables;`

注意:没有 `show views` 语句;

8. 视图与表的关系

视图是表的查询结果，自然表的数据改变了，影响视图的结果

视图是表的查询结果，自然表的数据改变了，影响视图的结果。

(1) 视图的数据与表的数据一一对应时，可以修改。

(2) 视图增删该也会影响表，但是视图并不是总是能增删该的。

```
create view lmj as select cat_id,max(shop_price) as lmj from goods group by cat_id;
```

```
mysql> update lmj set lmj=1000 where cat_id=4;
```

```
ERROR 1288 (HY000): The target table lmj of the UPDATE is not updatable
```

(3) 对于视图insert还应注意，视图必须包含表中没有默认值的列。

注意：一般来说，视图只是用来查询的，不应该执行增删改的操作

9. 视图算法

Algorithm = merge/temptable/updefined

Merge:当引用视图时，引用视图的语句与定义视图的语句合并(默认)

Temptable:当引用视图时，根据视图的创建语句建立一个临时表

Undefined: 未定义，自动让系统帮你选。

Merge:意味着，视图知识一个语句规则，当查询视图时，把查询视图的语句(比如 where 那些)与创建时的语句 where 子句等合并，分析，形成一条 select 语句。

Temptable:是根据创建语句瞬间创建一张临时表，然后查询视图的语句，从该临时表查数据。

在创建视图时的语句:where shop_price>1000;l

查询视图时，where shop_price<3000;

#那么查此视图时，真正发生的是 where(select where) and (view where)

二、SQL 编程

1、变量声明

(1) 会话变量

定义形式:

Set @变量名 = 值;

说明:

1. 跟 php 类似，第一次给其赋值，就算定义了
2. 它可以在编程环境和非编程环境中使用!
3. 使用的任何场合也都带该"@”符号。

(2) 普通变量

定义形式:

Declare 变量名 类型 【default 默认值】;

说明:

1. 它必须先声明(即定义)，此时也可以赋值;
2. 赋值跟会话变量一样:set 变量名 = 值;
3. 它只能在编程环境中使用!!!

说明:什么是编程环境?

存储过程，函数，触发器

(3) 变量赋值形式

语法 1:

Set 变量名 = 表达式; #语法中的变量必须先使用 declare 声明

语法 2:

Set @变量名 = 表达式;

此方式可以无需 declare 语法声明，而是直接赋值，类似 php 定义变量并赋值。

```
mysql> set @num = 45+45;
Query OK, 0 rows affected (0.00 sec)

mysql> select @num;
+-----+
| @num |
+-----+
|    90 |
+-----+
1 row in set (0.00 sec)
```

语法 3:

Select @变量名:= 表达式;

此语句会给该变量赋值，同时还会作为一个 select 语句输出“结果集”。

```
mysql> select @num:=45+45;
+-----+
| @num:=45+45 |
+-----+
|          90 |
+-----+
1 row in set (0.00 sec)
```

语法 4:

Select 表达式 into @变量名;# 此语句虽然看起来是 select 语句,但其实并不输出"结果集",而是给变量赋值。

```
mysql> select 45+45 into @num;
Query OK, 1 row affected (0.00 sec)

mysql> select @num;
+-----+
| @num |
+-----+
|    90 |
+-----+
1 row in set (0.00 sec)
```

2、运算符

(1) 算术运算符

+, -, *, /, %

注意:mysql 没有++和—运算符

(2) 关系运算符

>, >=, <, <=, =(等于), <>(不等于)!!

(3) 逻辑运算符

And (与)、or (或)、not(非)

3、语句块包含符

所谓语句块包含符,在 js 或 php 中,以及绝大部分的其他语言中,都是大括号: {} 它用在很多场合: if,switch,for,function

而 mysql 编程,语句块包含符是。

```
[begin_label:] BEGIN
    [statement_list]
END [end_label]
```

标识符,可省略,但如果在该结构中需要退出,则需要它。前后必须是同名的。

begin...end结构相当于php或js语言中的大括号语法,可以用在很多场合,比如if, case, while, loop等。

中的

4、If 判断

MySQL 支持两种判断，第一个是 if 判断，第二个 case 判断

If 语法

单分支

If 条件 then

// 代码

End if;

双分支

If 条件 then

 代码 1

Else

 代码 2

End if;

多分支

If 条件 then

 代码 1

Elseif 条件 then

 代码 2

Else

 代码 3

End if;

案例：接收 4 个数字，

如果输入 1 则输出春天，2=>夏天，3=>秋天，4=>冬天 其他数字=>出错

我们使用存储过程来体验 if 语句的用法，

CREATE procedure 存储过程名(参数,参数,...)

BEGIN

//代码

END;

注意:通常情况下,以";"表示 SQL 语句结束,同时向服务器提交并执行。但是存储过程中有很多 SQL 语句,每一句都要以分号隔开,这时候我们就需要使用其他符号来代替向服务器提交的命令。通过 delimiter 命令更改语句结束符。

```
mysql> delimiter $
mysql> select uid,nickname,gender from user limit 3$
```

uid	nickname	gender
200050	胖胖	0
200051	刘林嵩	1
200052	小松鼠	0

```
3 rows in set (0.00 sec)
```

具体的语句:

Create procedure p1(n int)

Begin

If n =1 then select '春天' as '季节';

Elseif n = 2 then select '夏天' as '季节';

Elseif n = 3 then select '秋天' as '季节';

Else n = 4 then select '冬天' as '季节';

Else

Select '无法五天' as '季节';

Endif;

End\$

```
mysql> create procedure p1 (n int)
-> begin
-> if n=1 then
-> select '春天' as '季节';
-> elseif n=2 then
-> select '夏天' as '季节';
-> elseif n=3 then
-> select '秋天' as '季节';
-> elseif n=4 then
-> select '冬天' as '季节';
-> else
-> select '无法无天' as '季节';
-> end if;
-> end$
Query OK, 0 rows affected (0.01 sec)

mysql>
```

调用:语法: call 存储过程的名称(参数)

```
mysql> call p1 (3)$
+-----+
| 季节 |
+-----+
| 秋天 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

5、case 判断

Case 变量

When 值 then 语句;

When 值 then 语句;

Else 语句;

End case;

Create procedure p2(n int)

Begin

Case n

When 1 then select '春天' as '季节';

When 2 then select '夏天' as '季节';

When 3 then select '秋天' as '季节';

When 4 select '冬天' as '季节';

Else select '无法无天' as '季节';

End case;

End\$

```
mysql> create procedure p2 (n int)
-> begin
-> case n
-> when 1 then select '春天' as '季节';
-> when 2 then select '夏天' as '季节';
-> when 3 then select '秋天' as '季节';
-> when 4 then select '冬天' as '季节';
-> else select '无法无天' as '季节';
-> end case;
-> end$
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> call p2(23)$
+-----+
| 季节 |
+-----+
| 无法无天 |
+-----+
1 row in set (0.00 sec)
```

6、循环

MySQL 支持的循环有 loop、while、repeat 循环

(1) loop 循环

标签名:

Loop

Leave 标签名 – 退出循环

End loop;

案例:使用 loop 循环,完成计算 1 到 n 的和;

```

1  -- 计算1到n的和
2  create PROCEDURE p3(n INT)
3  BEGIN
4  declare i int default 1;
5  declare s int default 0;
6  aa:LOOP
7  if i > n then
8  leave aa;
9  end if;
10 set s = s+i;
11 set i = i+1;
12 end loop;
13 select s;
14 end;

```

(2) while 循环

[标签:] while 条件 do

// 代码

End while;

```

1  -- 计算1到n的和
2  create PROCEDURE p4(n int)
3  begin
4  declare i int default 1;
5  declare s int default 0;
6  while i <= n do
7  set s = s + i;
8  set i = i + 1;
9  end while;
10 select s;
11 end

```

(3) repeat 循环

Repeat

// 代码

Until 条件 end repeat;

```

2  -- 计算1到n的总和
3  create procedure p5(n int)
4  begin
5  declare i int default 1;
6  declare s int default 0;
7  repeat
8  set s = s + i;
9  set i = i + 1;
10 until i > n
11 end repeat;
12 select s;
13 END

```


三、存储过程

1、概念

声明是存储过程

存储过程(procedure)

概念类似于函数，就是把一段代码封装起来，当要执行这一段代码的时候，可以通过调用该存储过程来实现。在封装的语句体里面，可以同 if/else,case,while 等控制结构。可以进行 sql 编程。

查看现有的存储过程。

Show procedure status

2、存储过程的优点

存储过程 (Stored Procedure) 是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象，任何一个设计良好的数据库应用程序都应该用到存储过程。

(1) 存储过程只在创建时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次，所以使用存储过程可提高数据库执行速度。

(2) 当对数据库进行复杂操作时(如对多个表进行 Update,Insert,Query,Delete 时)，可将此复杂操作存储过程封装起来与数据库提供的事务处理结合一起使用。

(3) 存储过程可以重复使用,可减少数据库开发人员的工作量

(4) 安全性高,可设定只有某些用户才具有对指定存储过程的使用权

3、创建存储过程

Create procedure 存储过程名(参数,参数,...)

Begin

//代码

End

存储过程的参数分为输入参数(in)、输出参数(out)、输入输出参数(intout)，默认是输入参数。如果存储过程中就一条语句，begin 和 end 是可以省略的。

说明:

- (1) 存储过程中,可有各种编程元素:变量,流程控制,函数调用;
- (2) 还可以有:增删改查等各种 mysql 语句;
- (3) 其中 select(或 show,或 desc)会作为存储过程执行后的结果集返回
- (4) 形参可以设定数据的进出方向;

案例 1: 查询一个表里面某些语句

```

1 create procedure p10()
2 BEGIN
3 select goods_id,goods_name,shop_price from ecs_goods;
4 END

```

调用结果:

```

mysql> call p10()$
+-----+-----+-----+
| goods_id | goods_name          | shop_price |
+-----+-----+-----+
| 1 | KD876              | 1388.00 |
| 4 | 诺基亚N85原装充电器 | 58.00 |
| 3 | 诺基亚原装5800耳机 | 68.00 |
| 5 | 索爱原装M2卡读卡器 | 20.00 |
+-----+-----+-----+

```

案例 2: 第二个存储过程体会参数

输入一个字符串, 比如 h, 如果大于 h, 则取出价格大于 1000 的商品, 输入其他的值, 就输出小于 1000 的值的商品

```

1 create procedure p11(str varchar)
2 begin
3 if str > h then
4 select goods_id,goods_name,shop_price from ecs_goods where shop_price>1000;
5 else
6 select goods_id,goods_name,shop_price from ecs_goods where shop_price<=1000;
7 end if;
8 END

```

4、调用存储过程

语法: call 存储过程()

Mysql_query("call 存储过程名称()");

5、删除存储过程

语法: drop procedure [if exists] 存储过程名

6、创建复杂的存储过程

案例 3: 带输出参数的存储过程

```
create procedure p12(in n int,out res int)
begin
    set res = n*n;
end$

mysql> set @res=0;call p12(100,@res);select @res$
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

+-----+
| @res |
+-----+
| 10000 |
+-----+
```

案例 4：带有输入输出参数的存储过程

```
1 -- 带有输入输出参数的存储过程
2 create procedure p13(inout n int)
3 BEGIN
4     set n = n*n;
5 END

1 set @a = 20;call p13(@a); select @a;
```

@a
400

7、declare 声明局部变量

- (1) declare 声明局部变量的
- (2) 语法：declare 变量名数据类型[default value];
- (3) 通过 select 字段 into 变量或 set 给变量赋值
- (4) 变量名不可以和表的字段名一样

8、用户变量

用户变量只要在前面加一个“@”符号即可

Set @name = ‘李白’;

Select @name;

9、系统变量

MySQL 启动的时候就存在的变量，以@@开头的都是系统变量

Select @@version;

```
mysql> select @@version;
+-----+
| @@version |
+-----+
| 5.5.40    |
+-----+
1 row in set (0.00 sec)
```

四、函数

1、自定义函数

它跟 php 或 js 中的函数几乎一样：需要先定义，然后调用（使用）。

只是规定，这个函数，必须要返回数据-----要有返回值

（1） 定义语法：

Create function 函数名(参数) returns 返回值类型

Begin

// 代码

End



说明：

- (1) 函数内部可以有各种编程语言的元素：变量，流程控制，函数调用；
- (2) 函数内部可以有增删改等语句！
- (3) 但：函数内部不可以有 select(或 show 或 desc)这种返回结果集的语句！

（2） 调用

跟系统函数调用一样：任何需要数据的位置，都可以调用该函数。

案例 1：返回两个数的和

```

1 # 自定义函数 返回两个数的和
2 create function fun1(a int, b int) returns INT
3 begin
4     -- 代码
5     return a + b;
6 end

```

```

1 select fun1(100,120);

```

fun1(100,120)
220

案例 2：定义一个函数，返回 1 到 n 的和

```

1 -- 从1到n的和的函数
2 create function fun2(n int) returns int
3 begin
4     declare i int default 1;
5     declare s int default 0;
6     while i<= n do
7         set s = s + i;
8         set i = i + 1;
9     end while;
10    return s;
11 END

```

```

1 select fun2(100) as '和'

```

和
5050

注意：创建的函数，是隶属于数据库的，只能在创建函数的数据库中使用。

```

mysql> use whshop$
Database changed
mysql> select he(120,79)$
ERROR 1305 (42000): FUNCTION whshop.he does not exist
mysql>

```

2、系统函数

(1) 数字类

```
mysql> select rand();//返回 0 到 1 间的随机数
mysql> select * from it_goods order by rand() limit 2;//随机取出 2 件商品
mysql> select floor(3.9)//输出 3
mysql> select ceil(3.1)//输出 4
mysql> select round(3.5)//输出 4 四舍五入
```

(2) 大小写转换

```
mysql> select ucase('I am a boy!')// --转成大写
mysql> select lcase('I am a boy!')// --转成小写
```

(3) 截取字符串

```
mysql> select left('abcde',3)// --从左边截取
mysql> select right('abcde',3) // --从右边截取
mysql> select substring('abcde',2,3)// --从第二个位置开始，截取 3 个，位置从 1 开始
mysql> select concat(10,'锄禾日当午')// --字符串相连
mysql> select coalesce(null,123);
```

coalesce(str1,str2): 如果第 str1 为 null, 就显示 str2

```
mysql> select stuname,stusex,coalesce(writtenexam,'缺考'), coalesce(labexam,'缺考') from
stuinfo natural left join stumarks//
mysql> select length('锄禾日当午')// 输出 10 显示字节的个数
mysql> select char length('锄禾日当午') // 输出 5 显示字符的个数
```

(4) 时间类

函数名称	描述
ADDDATE()	相加日期
ADDTIME()	相加时间
CONVERT_TZ()	从一个时区转换到另一个时区
CURRENT_DATE,CURRENT_DATE	CURDATE()函数的同义词
CURRENT_TIME(),CURRENT_TIME	CURTIME()函数的同义词
CURRENT_TIMESTAMP()	NOW()函数的同义词
CURTIME()	返回当前时间
DATE_ADD()	两个日期相加
DATE_FORMAT()	按格式指定日期
DATE_SUB()	两个日期相减
DATE()	提取日期或日期时间表达式日期部分
DATEDIFF()	两个日期相减
DAYNAME()	返回星期的名字
DAY()	DAYOFMONTH()函数的同义词

DAYOFMONTH()	返回该月的第几天(1-31)
Dayofweek()	返回参数的星期索引
Dayofyear()	返回一年中的天(1-366)
Extract	提取日期部分
From_days()	日期的数字转换为一个日期
From_unixtime()	格式化日期为 UNIX 时间戳
Hour()	提取小时部分
Last_day	返回该参数对应月份的最后一天
Localtime(),localtime	NOW() 函数的同义词
Localtimestamp,localtimestamp()	NOW() 函数的同义词
Makedate()	从一年的年份和日期来创建日期
Microsecond()	从参数返回微妙
Maketime	Maketime()
Minute()	从参数返回分钟
Month()	通过日期参数返回月份
Monthname()	返回月份的名称
Now()	返回当前日期和时间
Period_add()	添加一个周期到一个年月
Period_diff()	返回两个时期之间的月数
Quarter()	从一个日期参数返回季度
Sec_to_time()	转换为“HH:MM:SS”的格式
Second()	返回秒(0-59)
Str_to_date()	转换一个字符串为日期
Subdate()	当调用三个参数时，它就是 date_sub
Subtime()	相减时间
Sysdate()	返回函数执行时的时间
Time_format()	格式化为时间
Time_to_sec()	将参数转换成秒并返回
Time()	提取表达式传递的时间部分
Timediff()	相减时间
Timestamp()	带一个参数,这个函数返回日期或日期表达式
Timestampadd()	添加一个时间间隔到 datetime 表达式
Timestampdiff()	从日期时间表达式减去的间隔
To_days()	返回日期参数转换为天
Unix_timestamp()	返回一个 UNIX 时间戳
Utc_date()	返回当前 UTC 日期
Utc_time()	返回当前 UTC 时间
Utc_timestamp()	返回当前 utc 日期和时间
Week()	返回周数
Weekday()	返回星期的索引
Weekofyear()	返回日期的日历周(1-53)
Year()	返回年份
Yearweek()	返回年份和周


```
mysql> select unix_timestamp();// --时间戳
mysql> select from_unixtime(unix_timestamp()) // --将时间戳转成日期格式
curdate();返回今天的时间日期:
mysql> select now();// --取出当前时间
```

```
mysql> select from_unixtime(unix_timestamp(), '%Y-%m-%d');
+-----+
| from_unixtime(unix_timestamp(), '%Y-%m-%d') |
+-----+
| 2016-12-12 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> select year(now()) 年, month(now()) 月, day(now()) 日, hour(now()) 小时, minute(now())
分钟, second(now()) 秒//
mysql> select datediff(now(), '1997-7-1') // 两个日期相距多少天
if(表达式, 值 1, 值 2): 类似于三元运算符
mysql> select concat(10, if(10%2=0, '偶数', '奇数'))//
```

if(表达式, 值 1, 值 2): 类似于三元运算符

```
mysql> select concat(10, if(10%2=0, '偶数', '奇数'))//
```

案例 1: 比如一个电影网站, 求出今天添加的电影; 在添加电影时, 有一个添加的时间戳。

//curdate()求出今天的日期

//把添加的时间戳, 转换成日期

```
Select title from dede_archives where curdate()=from_unixtime(senddate, '%Y-%m-%d');
```

```
mysql> select title from dede_archives where curdate()=from_unixtime(senddate, '%Y-%m-%d')$
+-----+
| title |
+-----+
| 叶问 |
+-----+
1 row in set (0.00 sec)
```

案例 2: 比如一个电影网站, 求出昨天添加的电影; 在添加电影时, 有一个添加的时间戳。

思路:

把添加的时间戳, 转换成日期, 和昨天的日期比较,

问题？如何求出昨天的日期，

扩展，如何取出昨天或者指定某个时间的电影：

Date_sub 和 date_add 函数：

基本用法：

Date_sub(时间日期时间,interval 数字 时间单位)

说明：

- (1) 时间单位：可以是 year,month,day hour,minute, second
- (2) 数字：可以是正数和负数

比如：取出昨天的日期：

Select date_sub(curdate(),interval 1 day);

```
mysql> Select date_sub(curdate(),interval 1 day);
+-----+
| date_sub(curdate(),interval 1 day) |
+-----+
| 2016-12-11                          |
+-----+
1 row in set (0.00 sec)
```

1、Select title from dede_archives where

datediff(curdate(),from_unixtime(senddate,'%Y-%m-%d')) = 1;

2、Select title from dede_archives where date_sub(curdate(),interval 1 day) =

from_unixtime(senddate,'%Y-%m-%d');

3、select title from dede_archives where date_add(curdate(), interval -1 day) =

from_unixtime(senddate,'%Y-%m-%d');

比如：取出前天的日期：

select date_sub(curdate(),interval 2 day)

或

select date_add(curdate(),interval -2 day)

```
mysql> select title from dede_archives where date_add(curdate(),interval -9 day)=from_unixtime(senddate,'%Y-%m-%d')$
```

title
新八卦莲花掌
新游侠黑蝴蝶
李连杰亲
中天
绝户先生
僵尸先生
风云决
花落花开
新英雄
老炮儿

```
10 rows in set (0.00 sec)
```

五、触发器

1、简介

(1) 触发器是一个特殊的存储过程，它是 MySQL 在 insert、update、delete 的时候自动执行的代码块。

(2) 触发器必须定义在特定的表上。

(3) 自动执行，不能直接调用，

作用：监视某种情况并触发某种操作。

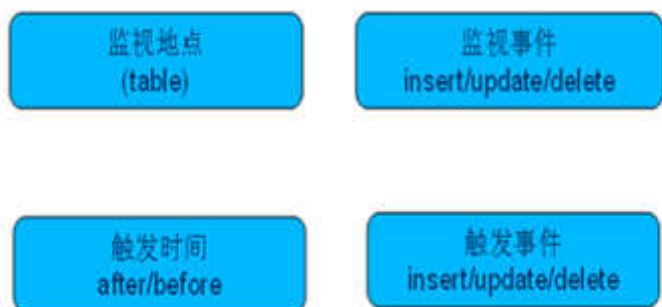
触发器的思路：

监视 it_order 表，如果 it_order 表里面有增删改的操作，则自动触发 it_goods 里面增删改的操作。

比如新添加一个订单，则 it_goods 表，就自动减少对应商品的库存。

比如取消一个订单，则 it_goods 表，就自动增加对应商品的库存减少的库存。

2、触发器四要素



3、创建触发器

创建触发器的语法：

Create trigger trigger_name

After/before insert/update/delete on 表名

For each row

Begin

Sql 语句(触发的语句一句或多句)

End

案例 1：第一个触发器，购买一头猪，减少 1 个库存

分析：

监视的地点：it_order 表

监视的事件：it_order 表的 insert 操作。

触发的时间：it_order 表的 insert 之后

触发的事件 it_goods 表减少库存的操作。

Create trigger t1

After insert on it_order

For each row

Begin

Update it_goods set goods_number = goods_number - 1 where id = 1;

End

注意:如果在触发器中引用行的值。对于 Insert 而言，新增的行用 new 来表示，行中的每一列的值，用 new.列名来表示。

```

+-----+
| 1 | 烧饼 | 22 |
| 2 | 馒头 | 22 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 22 |
+-----+
4 rows in set (0.00 sec)

mysql> select * from it_order$
Empty set (0.01 sec)

mysql> desc it_order$
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| order_id | int(11) | YES | | NULL | |
| goods_id | int(11) | YES | | NULL | |
| much | int(11) | YES | | NULL | |
+-----+
3 rows in set (0.04 sec)

mysql> insert into it_order values(1,2,1)$
Query OK, 1 row affected (0.01 sec)

mysql> select * from it_goods$
+-----+
| id | goods_name | goods_number |
+-----+
| 1 | 烧饼 | 22 |
| 2 | 馒头 | 21 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 22 |
+-----+
4 rows in set (0.00 sec)

```

上面我们创建的触发器 t1 是有问题的，我们购买任何商品都是减少馒头的库存

案例 2：购买商品，减少对应库存

Create trigger t2

After insert on it_order

For each row

Begin

Update it_goods set goods_number = goods_number - new.much where id = new.goods_id;

End

```
mysql> select * from it_goods$
+-----+-----+-----+
| id    | goods_name | goods_number |
+-----+-----+-----+
| 1    | 烧饼      | 22          |
| 2    | 馒头      | 20          |
| 3    | 肉饼      | 22          |
| 4    | 包子      | 22          |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from it_order$
Empty set (0.00 sec)

mysql> insert into it_order value(1,1,5)$
Query OK, 1 row affected (0.01 sec)

mysql> select * from it_goods$
+-----+-----+-----+
| id    | goods_name | goods_number |
+-----+-----+-----+
| 1    | 烧饼      | 17          |
| 2    | 馒头      | 20          |
| 3    | 肉饼      | 22          |
| 4    | 包子      | 22          |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

案例 3：取消订单时，减掉的库存要添加回来

```
1 -- 取消订单时，减掉的库存要添加回来。
2 -- 分析：
3 -- 监视地点：it_order表
4 -- 监视的事件：it_order表的delete操作
5 -- 触发的时间：it_order表的delete操作之后
6 -- 触发的事件：it_goods表，把减掉的库存恢复起来
7 create trigger t3
8 after DELETE on it_order
9 for each row
10 BEGIN
11     update it_goods set goods_number=goods_number+old.much where id=old.goods_id;
12 END
13
14 -- 对于it_order表，删除的行我们使用old来表示，如果要引用里面的数据，
15 -- 则使用old.列名来表示。
```

```

mysql> select * from it_goods$
+-----+-----+-----+
| id | goods_name | goods_number |
+-----+-----+-----+
| 1 | 烧饼 | 17 |
| 2 | 馒头 | 20 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 22 |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from it_order$
+-----+-----+-----+
| order_id | goods_id | much |
+-----+-----+-----+
| 1 | 1 | 5 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> delete from it_order where order_id=1$
Query OK, 1 row affected (0.01 sec)

mysql> select * from it_goods$
+-----+-----+-----+
| id | goods_name | goods_number |
+-----+-----+-----+
| 1 | 烧饼 | 22 |
| 2 | 馒头 | 20 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 22 |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

注意：如果在触发器中引用行的值。

对于 insert 而言，新增的行用 new 来表示，行中的每一列的值，用 new.列名来表示。

注意:目前 mysql 不支持多个具有同一个动作，同一时间，同一事件，同一地点的触发器

it_order 表里面删除的行我们使用 old 来表示，行中的列用 old.字段名称来表示。

案例 4：修改订单时，库存也要做对应修改（修改购买数量）

- 修改订单时，库存也要做对应修改
- 监视的地点: it_order 表
- 监视的事件: it_order 表的 update 操作
- 触发的时间: it_order 表的 update 操作之后
- 触发的事件: it_goods，要修改对应的库存

Create trigger t4

After update on it_order

For each row

Begin

Update it_goods set goods_number=goods_number+old.much where
goods_id=old.goods_id;

Update it_goods set goods_number=goods_number-new.much where
goods_id=new.goods_id;

End

-- 完成修改的思路:

-- (1) 撤销订单。It_goods 表里面的库存要回复

-- (2) 重新下订单。It_goods 表里面的库存要减少

```
mysql> select * from it_order$
+-----+-----+-----+
| order_id | goods_id | much |
+-----+-----+-----+
|         1 |         1 |     5 |
+-----+-----+-----+
```

```
mysql> select * from it_goods;
```

id	goods_name	goods_number
1	猪	15
2	羊	13
3	狗	3
4	猫	22
5	驴	22
6	马	45

```
6 rows in set (0.00 sec)

mysql> truncate it_order;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into it_order values(1,1,5);
Query OK, 1 row affected (0.00 sec)

mysql> select * from it_goods;
```

id	goods_name	goods_number
1	猪	10
2	羊	13
3	狗	3
4	猫	22
5	驴	22
6	马	45

```
6 rows in set (0.00 sec)
```

```
mysql> delete from it_order where order_id = 1;
Query OK, 1 row affected (0.00 sec)

mysql> select * from it_goods;
```

id	goods_name	goods_number
1	猪	15
2	羊	13
3	狗	3
4	猫	22
5	驴	22
6	马	45

```
6 rows in set (0.00 sec)

mysql>
```

注意：如果是修改操作，要引用 `it_order` 表里面的值，

修改前的数据，用 `old` 来表示，`old.列名` 引用被修改之前行中的值。

修改后的数据，用 **new** 来表示，**new.列名** 引用被修改之后行中的值

4、删除触发器

语法：drop trigger 触发器的名称

5、查看触发器

语法：show triggers

6、before 与 after 的区别

After 是先完成数据的增删改，再触发，触发器中的语句晚于监视的增删改，无法影响前面的增删改操作。就类似与先吃饭，再付钱。

Before 是先完成触发，在增删改，触发的语句先于监视的增删改发生，我们有机会判断修改即将发生的操作。就类似与先付钱，再吃饭。

典型案例：对于已下的订单，进行判断，如果订单的数量>5，就认为是恶意订单，强制把所定的商品数量改为 5

分析：

监视的地点：it_order 表

监视的事件:it_order 表的 insert 操作

触发的时间:it_order 表的 insert 操作之前

触发的事件:如果购买数量大于 5， 把购买数量改成 5

Create trigger t5

Before insert on it_order

For each row

Begin

If new.much>5 then

New.much = 5;

End if;

End;

```
mysql> select * from it_goods$
+-----+-----+-----+
| id | goods_name | goods_number |
+-----+-----+-----+
| 1 | 烧饼 | 22 |
| 2 | 馒头 | 15 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 12 |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from it_order$
+-----+-----+-----+
| order_id | goods_id | much |
+-----+-----+-----+
| 1 | 2 | 5 |
| 2 | 4 | 10 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> insert into it_order values(3,4,10)$
Query OK, 1 row affected (0.01 sec)

mysql> select * from it_goods$
+-----+-----+-----+
| id | goods_name | goods_number |
+-----+-----+-----+
| 1 | 烧饼 | 22 |
| 2 | 馒头 | 15 |
| 3 | 肉饼 | 22 |
| 4 | 包子 | 7 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

六、游标

假设公司为员工提供了基于出勤工资和佣金比率的假期奖金，但是奖金的数额取决于很多因素，范围如下表示：

如果工资>60 000 美元且佣金比率>5%，则奖金=工资*佣金比率

如果工资>60 000 美元且佣金避雷线<=5%，则奖金=工资*3%

对于所有其他员工，奖金=工资*7%

1. 创建游标

其形式如下：

```
DECLARE cursor_name CURSOR FOR select_statement;
```

例如：要声明前面讨论的奖金计算游标，可执行如下声明：

```
DECLARE calc_bonus CURSOR FOR SELECT id,salary,commission FROM employees;
```

声明游标后，必须打开才能使用。

2. 打开游标

虽然游标查询在 `DECLARE` 语句中定义，但在打开游标之前查询并未真正执行。可以使用 `OPEN` 语句完成此任务。

```
OPEN cursor_name
```

例如，为打开本节前面创建的 `calc_bonus` 游标，要执行：

```
OPEN calc_bonus;
```

3. 使用游标

使用游标指向的信息是通过 `FETCH` 语句完成的。其形式如下：

```
FETCH cursor_name INTO varname1[,varname2.....]
```

例如，以下存储过程 `calculate_bonus()` 获取游标指向的 `id,salary` 和 `commission` 字段，完成必要的比较，最后插入适当的奖金：

4. 关闭游标

在使用游标完毕之后，应当用 `CLOSE` 语句关闭它，回收重要的系统资源。为关闭打的游标 `calc_bonus`，执行：

```
CLOSE calc_bonus;
```

5. 游标实例

```
DELIMITER $$
CREATE PROCEDURE calculate_bonus()
BEGIN
    declare emp_id int;
    declare sal decimal(8,2);
    declare comn decimal(3,2);
    declare done int;

    declare calc_bonus cursor for select id,salary,commission from employees;

    declare continue handler for not found set done = 1;
```

```

open calc_bonus;

begin_calc:loop
    fetch calc_bonus into emp_id,sal,comm,
    if done then
        leave begin_calc;
    end if;

    if sal > 60000.00 then
        if comm > 0.05 then
            update employees set bonus = sal * comm where id = emp_id;
        else if comm <= 0.05 then
            update employees set bonus = sal * 0.03 where id = emp_id;
        end if;
    else
        update employees set bonus = sal * 0.07 where id = emp_id;
    end if;
end loop begin_calc;
close calc_bonus
END $$
DELIMITER;

```

七、MySQL 查询

-- sql 语句查询排名

-- 思路:有点类似循环里面的自增一样， 设置一个变量并赋予初始值， 循环一次自增加

1， 从而实现排序;

-- mysql 里则是需要将数据查询出来并先行按照小执行的字段做好降序 desc,或者升序 asc,设置好排序的变量(初始值为 0);

-- a>.将已经排序好的数据从第一条一次取出来， 取一条就自增加一， 实现从 1 到最后的一个排名

-- b>.当出现相同的数据时， 排名保持不变， 此时则需要再设置一个变量， 用来记录上一条数据的值， 跟当前数据的值进行对比， 如果相同， 则排名不变， 不相同则排名自增加 1

-- c>.当出现相同的数据时，排名保持不变，但是保持不变的排名依旧会占用一个位置，也就是类似于(1,2,2,2,5)这种排名就是属于中间的三个排名是一样的，但是第五个排名按照上面一种情况(1,2,2,2,3),现在则是排名相同也会占据排名的位置

-- 准备数据(用户 id,分数):

```
drop table if exists sql_rank;
```

```
CREATE TABLE sql_rank(  
    id int unsigned not null auto_increment,  
    user_id int unsigned not null,  
    score tinyint(3) unsigned not null,  
    add_time date not null,  
    primary key(id)  
)engine=innodb auto_increment=1 charset=utf8;
```

-- 插入数据

```
insert into sql_rank(user_id,score,add_time)  
  
VALUES  
  
(100,50,'2016-05-01'),  
(101,30,'2016-05-01'),  
(102,20,'2016-05-01'),  
(103,60,'2016-05-01'),  
(104,80,'2016-05-01'),  
(105,50,'2016-05-01'),  
(106,70,'2016-05-01'),  
(107,85,'2016-05-01'),  
(108,60,'2016-05-01');
```

当前数据库数据:

id	user_id	score	add_time
1	100	50	2016-05-01
2	101	30	2016-05-01
3	102	20	2016-05-01
4	103	60	2016-05-01
5	104	80	2016-05-01
6	105	50	2016-05-01
7	106	70	2016-05-01
8	107	85	2016-05-01
9	108	60	2016-05-01

2、sql1{不管数据相同与否，排名依次排序(1,2,3,4,5,6,7...)}

```
SELECT
    obj.user_id,
    obj.score ,@rownum :=@rownum + 1 AS rownum
FROM
    (
        SELECT
            user_id,
            score
        FROM
            `sql_rank`
        ORDER BY
            score DESC
    ) AS obj,
    (SELECT @rownum := 0) r
```

执行的结果如下图:

user_id	score	rownum
107	85	1
104	80	2
106	70	3
103	60	4
108	60	5
100	50	6
105	50	7
101	30	8
102	20	9

可以看到，现在按照分数从 1 到 9 都排好了，但是有些分数相同的用户排名却不一样，这就是接下来要说的第二种 sql

2、sql2{只要数据有相同的排名就一样，排名一次排序(1,2,2,3,3,4,5....)}

```

SELECT
    obj.user_id,
    obj.score,
    CASE
        WHEN @rowtotal = obj.score THEN
            @rownum
        WHEN @rowtotal := obj.score THEN
            @rownum := @rownum + 1
        WHEN @rowtotal = 0 THEN
            @rownum := @rownum + 1
        END AS rownum
FROM
    (
        SELECT
            user_id,
            score
        FROM
            `sql_rank`
        ORDER BY
            score DESC
    ) AS obj,
    (SELECT @rownum := 0 , @rowtotal := NULL) r

```

这时候就新增加了一个变量，用于记录上一条数据的分数了，只要当前数据分数跟上一条数据的分数比较，相同分数的排名就不变，不相同分数的排名就加一，并且更新变量的分数值为该条数据的分数，依次比较

如下图结果:

user_id	score	rownum
107	85	1
104	80	2
106	70	3
103	60	4
108	60	4
100	50	5
105	50	5
101	30	6
102	20	7

跟第一条 sql 的结果相对比你会发现，分数相同的排名也相同，并且最后一名名次由第 9 名变成第 7 名

如果你需要分数相同的排名也相同，但是后面的排名不能受到分数相同排名相同而不占位的影响，也就是哪怕你排名相同，你也占了这个位置(比如:1,2,2,4,5,5,7...)这种形式的，虽然片名有相同，但是你占位了，后续的排名根据占位来排)

3、sql2{只要数据有相同的排名就一样，但是相同排名也占位，排名一次排序(1,2,2,4,5,5,7....)}

此时需要增加一个变量，来记录排序的号码(自增)

```
SELECT
    obj_new.user_id,
    obj_new.score,
    obj_new.rownum
FROM
    (
        SELECT
            obj.user_id,
            obj.score,
            @rownum := @rownum + 1 AS num_tmp,
            @incnum := CASE
                WHEN @rowtotal = obj.score THEN
                    @incnum
```



```

        WHEN @rowtotal := obj.score THEN
            @rownum
        END AS rownum
    FROM
        (
            SELECT
                user_id,
                score
            FROM
                `sql_rank`
            ORDER BY
                score DESC
        ) AS obj,
        (
            SELECT
                @rownum := 0 ,@rowtotal := NULL ,@incrnum := 0
        ) r
    ) AS obj_new

```

上面 sql 执行的结果如下:

user_id	score	rownum
107	85	1
104	80	2
106	70	3
103	60	4
108	60	4
100	50	6
105	50	6
101	30	8
102	20	9

结果集中分数相同的，排名相同，同时也占据了那个位置，中间的一个数据过程

user_id	score	num_tmp	rownum
107	85	1	1
104	80	2	2
106	70	3	3
103	60	4	4
108	60	5	4
100	50	6	6
105	50	7	6
101	30	8	8
102	20	9	9

4、基本函数

阶乘函数

```
CREATE FUNCTION Factorial(n int) returns int
begin
    declare i int default 1; --
    declare result int default 1;
    while i<=n do
        set result = result*i;
        set i = i+1;
    end while;
    return result;
end
```

递增数列

```
create function ss(n int) returns int
begin
    declare i int default 1;
    declare s int default 0;
    while i<=n do
        set s = s+i;
        set i = i+1;
    end while;
    return s;
end
```

随机数字

```
create function getrand(n int) returns varchar(20)
begin
    declare s varchar(20);
```

```
declare counts int;
set s = round(round(rand(),n)*pow(10,n));
if char_length(s)<n then
    set counts = n - char_length(s);
    set s = concat(s,right(concat(pow(10,counts),"),counts));
end if;
if char_length(s)>n then
    set s = right(s,n);
end if;
return s;
end
```

MySQL 高效编程—学习笔记

一、MySQL 基础篇

1. 对于企业而言，选择 MySQL 数据库的两大原因：
 - 1) MySQL 是开源关系数据库产品，使用普及率高；
 - 2) 性能出色，运行速度快。MySQL 有免费和收费两种类型的产品
2. mysql 登录:cd 打开 mysql 的 bin 目录，执行”mysql -u root -p”回车，然后要求输入密码 Enter password, 正确输入密码后，回车，即可登录成功！

```

C:\Users\Administrator>cd D:\Program Files\phpStudy\MySQL\bin
C:\Users\Administrator>d:
D:\Program Files\phpStudy\MySQL\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.5.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

3. 常用数据库查询命令：

show databases; 查看数据库服务器上的全部数据库

create database test; 创建一个名称为 test 的数据库

use test; 使用 test 数据库

show tables; 查看当前数据库中所有表

创建表：

```

create table 表名(
    属性名数据类型列选项,
    .....
)

```

其中，列选项可以为 AUTO_INCREMENT(自增)，DEFAULT ‘默认值’，INDEX(定义为索引)，[NOT]

NULL(允许/禁止 NULL)，PRIMARY KEY(定义为主键)，UNIQUE(定义为唯一性)，CHECK(定义可以输入的值的范围/选项)。

alter table 表名 modify 列名 数据类型; 修改表中列的定义

alter table 表名 add 列名 数据类型; 追加新列

alter table 表名 drop 列名; 删除列

desc 表名; 显示表的结构

drop table 表名; 删除表

INSERT INTO 表名(列名 1, 列名 2, ……) values(数据 1, 数据 2, ……); 向表中插入数据

`select 列名 1, 列名 2, ... from 表名 [条件表达式等];` 显示表中的数据
`update 表名 set 列名 1=值 1, 列名 1=值 1, ... where 条件表达式;` 更新标志的记录
`delete from 表名 where 条件表达式;` 删除表中的记录

4. 数据检索

① 荐明确指定列名：使用[*]来无条件地获得所有列数据，第一，获取到不需要的列就会浪费很多内存；第二，获得的数据会按表定义的列的顺序，如果修改了表列的顺序，对应的程序也必须进行修改。

②条件检索：必须使用 WHERE 语句。比较运算符有：=、<、>、<=、>=、<>(不等于)、IS[NOT] NULL (是否为空)、[NOT] LIKE (指定目标一致)、[NOT] BETWEEN (指定范围)、[NOT] IN (在指定候补值内)。

③模糊检索：`select name from customer where name like '李%';`

④ 多条件组合查询 :使用 and(并)、or(或)连接。理论运算符的优先顺序 :NOT->AND->OR。

⑤结果排序：使用 order by 进行数据排序，可以指定 ASC (升序) 或 DESC (降序)，默认是 ASC。

如：`select name,age from customer where age>21 order by age DESC;`

⑥取得指定数间的记录：LIMIT ；

`limit 2`:从起始位置开始取出两件；

`limit 1,2`：从 1 开始取出两件。

⑦数据分组：GROUP BY，通常 GROUP BY 语句与统计函数一起使用。主要统计函数：

AVG(列名)平均值、COUNT(列名)件数、MAX(列名)最大值、MIN(列名)最小值、SUM(列名)合计值。

⑧列的别名：`select sex,COUNT(mid) as cnt from customer group by sex;`其中 as 可以省略。

5. 运算符与数据库函数

①运算符：`+`、`-`、`*`、`/`、`DIV`（除法返回结果的整数部分）、`%`（求余）、`AND`、`OR`。

②部分数据库函数

LENGTH 函数：返回字符串的字节数

FLOOR/CEILING/TRUNCATE 函数：用于小数四舍五入处理

DATE_ADD 函数：对日期 date 进行指定值的加算处理

NOW()当前时间、**CURDATE()**当前日期、**CURTIME()**当前时刻、**EXTRACT(YEAR_MONTH**
'2013-11-20 21:02:00')从给定的日期/时刻中取得任意元素（如年、月等）

CASE 函数：条件判断

```
[sql] 
01. select name
02.     case sex
03.         when 0 then '男'
04.         when 1 then '女'
05.         else 'OTH'
06.     end as sex
07. from customer;
```

6. 多表连接查询

内链接：返回结果集中仅是符合查询条件和连接条件的记录；

```
[sql] 
01. select m.name,o.oid
02.     from member m
03.     <span style="color:#ff0000;">inner join </span>ordercord o
04.     <span style="color:#ff6666;">on</span> m.id=o.mid
05.     (where/order by等语句)
```

外连接：返回的查询结果集中不仅是符合连接条件的行，而且还包括左表（左外连接）或右表（右外连接）中所有数据行；

左外连接：

```
[sql] 
01. select 列名1 from 表1
02.     LEFT OUTER JOIN 表2
03.     ON 表1.外键=表2.主键[WHERE/ORDER BY语句等]
```

右外连接：

```
[sql] 
01. select 列名1 from 表1
02.     RIGHT OUTER JOIN 表2
03.     ON 表1.外键=表2.主键[WHERE/ORDER BY语句等]
```

3 个或 3 个以上表间连接，从内往外一层层的连接。

③子查询：先返回子查询的结果集，再返回主查询结果集。

二、MySQL 高级应用篇

7、MySQL 的功能可以分为两部分：外层部分主要完成与客户端的连接以及事前调查 SQL 语句的内容的功能；而内层部分就是所谓的存储引擎部分，它负责接受外层的数据操作指示，完成实际的数据输入输出以及文件操作工作。MySQL 提供的存储引擎有：MyISAM（默认的高速引擎，不支持事务处理）、InnoDB（支持行锁定以及事务处理，比 MyISAM 的处理速度稍慢）、ISAM 等。

8、查看表中使用的引擎：`show create table 表名 [\G];`

修改表使用的引擎：`alter table 表名 engine=新引擎;`

其中，\G 在 MySQL 监视框中显示出信息有序。

====>事务

9、**事务**是指作为单个业务逻辑工作单元执行的一系列操作，要么全部成功，要么全部失败。

MySQL 提供的 3 个重要事务处理命令：BEGIN（声明事务处理开始）、COMMIT（提交整个事务）、ROLLBACK（回滚到事务开始的状态）。

10、MySQL 默认自动提交事务。当然也可以通过 `SET AUTOCOMMIT=0;`将自动提交功能置为 OFF；也可以通过 `SET AUTOCOMMIT=1;`将自动提交功能置为 ON。

11、部分回滚：

定义保存点：`savepoint 保存点名;`

回滚到指定的保存点：**rollback to savepoint 保存点名**；

12、事务处理的利用范围，以下几条 SQL 命令，执行后将被自动提交，是在事务处理可以利用的范围之外：DROP DATABASE；DROP TABLE；DROP；ALTER TABLE；

13、多用户同时操作同一个数据库时，需要对其进行并发控制，MySQL 通过锁来实现。锁分为**共享锁**和**排他锁**，共享锁是当用户参照数据时，将对象数据变为只读形式的锁定；排他锁是使用 INSERT/UPDATE/DELETE 命令对数据进行更新时，其他进程（或事务）一律不能对读取该数据。

14、锁定对象的大小，通常被称为锁定的**粒度**。支持的粒度随着数据库不同而有所差异，一般有 3 种锁定粒度：**记录（行）、表、数据库**；

15、锁定的粒度越小，运行性越高，但并不是越小的粒度越好，锁定的数目越多，消耗的服务器资源也越多。数据库中行单位粒度的锁定大量发生时，数据库有将这些锁定的粒度自动向上提升的机制，通常称为**锁定提升**。MySQL 只支持行/表粒度，不支持锁定提升功能。

16、事务处理的隔离级别：READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ、SERIALIZABLE。

隔离级别	非提交读取	不可重复读取	幻想读取（幻读）
READ UNCOMMITTED	V	V	V
READ COMMITTED	X	V	V
REPEATABLE READ	X	X	V
SERIALIZABLE	X	X	X

17、事务处理的机制简单的说就是留下更新日志，数据库会根据这些日志信息，在必要时将就数据取回，或在发生错误是将数据恢复到原先的状态。与事务处理相关的日志可以分为两个类型：UNDO 和 REDO。

====>>索引

18、索引就是表中所有记录的搜索引导，索引的经过排序的。有索引后，先查找索引再根据索引查找相应的记录。

19、大多数数据库中用 **B 树（平衡树）** 的结果来保存索引的。

20、创建索引：`create [unique] index 索引名 on 表名 (列名,.....)`;

查看表中所有的索引信息：`show index from 表名 [G]`;

删除索引：`drop index 索引名 on 表名`;

确认索引的使用情况（优劣）：`explain 调查对象 select 语句`;

如：`explain select * from employee where name='wang' \G`;

复合索引：`create index idx_employee_name on employee(name,sex);`

唯一索引：`create unique index idx_employee_name on employee(name);`

索引名建议[idx_]的形式开头。

21、不适合使用索引的场合：

①模糊查询要求后方一致或部分一致检索 ;如 :`select * from employee where name like '%w%'`或`'%w'`;是不适合的。

②使用了 IS NOT NULL、<>比较运算符的场合 ,如 :`select * from employee where name is not null`;是不适合的。

③对列使用了运算/函数的场合 ;

④复合索引的第一列没被包含在 where 条件语句中的场合。如 :`select * from employee where name='wang' or age='1'`;是不适合的。

=====>>视图

22、视图是伪表，视图本身不包含任何数据的，将多个物理表的数据通过视图动态地组织在一起，用户可以像使用普通物理表那样使用它。

23、视图具有以下特性：①可以公开表中特定的行和列；②简化复杂的 SQL 查询；③可以限制可插入/更新的值范围。

24、创建视图：`create view 视图名(列名,...) as select 语句 [with check point]`；

删除视图：`drop view 视图名`;

查看视图所有列信息：`show fields from 视图名`;

视图名建议[v_]的形式开头。

25、不能进行插入/更新/删除操作的场合：

①视图列中含有统计函数

②视图定义使用了 GROUP BY/HAVING 语句，DISTINCT 语句，UNION 语句的情况

注：HAVING 是对 group by 设置查询条件；distinct 是要求查询结果没有重复项。

③视图定义时使用了子查询

④跨越多个表进行数据的变更操作

26、创建视图时使用[with check option]命令，将不能插入或更新不符合视图的检索条件的数据。

如：

```
[sql]  [ ]  [ ]  [C]  [P]
01.  create view v_product300up
02.      as
03.      select * from product where price >=300
04.      with check option;
```

此时，变更操作要求 price 大于或等于 300 才能进行，否则就会出现[CHECK OPTION failed.....]的错误信息。

====>>存储过程

27、存储过程是数据库中保存的一系列 SQL 命令的集合，这些 SQL 命令通常并非简单的组合在一起，还可以使用各种条件判断、循环控制等，来实现简单的 SQL 命令不能实现的负责功能。

28、使用存储过程可以有这些好处：

①提高执行性能

②可减轻网络负担

③可防止对表的直接访问，提高安全性

④可将数据库的处理黑匣子化

29、定义存储过程：

```
create procedure 存储过程名( 参数的种类 1 参数 1 参数类型 1 [参数的种类 2 ,参数 2 ,  
参数类型 2...])
```

```
begin
```

```
处理内容
```

```
end
```

注：存储过程的参数可以分为输入参数（接受调用方的数据），输出参数（向调用方返回处理结果）。参数的种类可以是 IN、OUT、INOUT 其中之一。存储过程名前加上[sp_]的开头。

如：



```
[sql]     
01. <span style="color:#ff0000;">delimiter //</span>
02. create procedure sp_search_customer(IN p_name varchar(20) )
03. <span style="white-space:pre"> </span>begin
04. <span style="white-space:pre"> </span>if p_name is null or p_name='' then
05.   <span style="white-space:pre"> </span>select * from customer;
06. <span style="white-space:pre"> </span>else
07. <span style="white-space:pre"> </span> select * from customer where name like p_name;
08. <span style="white-space:pre"> </span>end if;
09. <span style="white-space:pre"> </span>end
10. <span style="color:#ff0000;">delimiter ;</span>
```

注：上面 delimiter 命令改变分隔符为//，因为在存储过程中使用到";"。

30、存储过程中可使用的控制语句：

①if...elseif...else...end if

②case when 值 1 then 执行命令 when 值 2 then 执行命令 else 执行名 end case

③循环控制（后置判断）：repeat 直至条件表达式为 true 时才执行的命令 until 条件表达式 end repeat

④循环控制（前置判断）：while 条件表达式 do 系列命令 end while

31、查看存储过程是否存在：show procedure status [G];

查看存储过程信息：show create procedure 存储过程名;

删除存储过程：drop procedure 存储过程名;

执行存储过程名：call 存储过程名（参数,.....）；

32、定义存储过程可以声明局部变量：declare 变量名 数据类型[初始值]；

赋值给变量：set 变量名=值；

=====>>存储函数

33、存储函数是保存在数据库中的函数（Function）。所谓函数就是按照事先决定的规则进行处理，然后将结果返回的单功能机制。

34、定义存储函数：

create function 函数名（

参数 1 数据类型 1

[, 数据 2 数据类型 2...]

)returns 返回值类型

begin

任意系列 SQL 语句

return 返回值；

end

注：①参数只有输入型 IN

②向调用返回结果值


③存储函数给定[fn_]的前缀

35、查看全部已创建的存储函数：`show function status;`

查看存储函数的信息：`show create function 函数名 [\G];`

存储函数调用：`select 函数名(参数);`

如：

```
[sql] 
01. delimiter //
02. create function fn_factorial(
03.     f_num int
04. )returns int
05. begin
06.     declare f_result int default 1;--定义局部变量并初始化为1
07.     while f_num>1 do
08.         set f_result=f_result*f_num;
09.         set f_num=f_num-1;
10.     end while;
11.     retrun f_result;
12. delimiter ;
```

调用存储函数：

```
[sql] 
01. select fn_factorial(5), fn_factorial(0);
```

=====>>触发器

36、触发器是在操作数据库时，执行一个动作，而触发了另一个动作。如删除表 1 的一条记录，在表 2 中插入一条日志记录。

37、创建触发器：

```
create trigger 触发器名 发生时刻 事件名
```

```
on 表名 for each row
```

```
begin
```

```
任意系列 SQL 语句
```

```
end
```

说明：①触发器不是直接与运行的，而是针对具体表的操作事时被调用的；

②决定触发器运行时刻具体是指发生在 insert、update、delete 等操作的前还是后，用 before 或 after 表示。

③for each row 表示以行为单位执行的，为固定值。

确认已创建完成的触发器列表和触发器的信息：`show trigger [G];`

删除触发器：`drop trigger 触发器名；`

```

[sql]
01. delimiter //
02. create trigger trg_customer_history after delete(
03.     on customer for each row
04.     begin
05.         insert into customer_history(mid,name,birth,sex,updated)
06.         <span style="white-space:pre">         </span>values(OLD.mid,OLD.name,OLD.birth,OLD.sex,NOW());
07.     end;
08. delimiter ;

```

====>>游标

38、游标就是对 select 语句取出的结果进行一件一件处理的功能。游标可以将多个查询记录进行一件一件的单独的处理。

```
CREATE PROCEDURE sp_cursor (OUT p_result text) BEGIN -- 定义标志变量 flag(判断是否所有记录都被取出)
```

```
DECLARE flag bit DEFAULT 0;
```

```
-- 定义存储当前行的部署名的变量 tmp
```

```
DECLARE tmp VARCHAR (20);
```

```
-- 定义游标取出游标中所有记录时的处理
```

```
DECLARE cur CURSOR FOR SELECT DISTINCT depart FROM employee;
```

```
-- 定义取出游标中所有记录是的处理
```

```
DECLARE CONTINUE HANDLER FOR NOT found SET flag = 1;
```

```
-- 打开游标
```

```
OPEN cur;
```

```
-- 从游标中一行行取出数据

WHILE flag != 1 DO FETCH cur INTO tmp;

-- 将当前行中的内容保存到本地变量中

IF flag != 1 THEN -- 将变量 tmp 以逗号分隔的字符串的形式保存到输出参数 p_result 中

SET p_result = concat_ws(',', p_result);

END IF;

END WHILE;

-- 标志变量为 1 后结束循环

-- 关闭游标

CLOSE cur;

END

调用游标 : call sp_cursor(@p_result);
```

三、Shell 高级编程企业实战题以及参考答案

1、监控 MySQL 主从同步

企业面试题 1：监控 MySQL 主从同步是否异常，如果异常，则发送短信或者邮件给管理员。提示：如果没有主从同步环境，可以用下面文本放到文件里读取来模拟；

阶段 1：开发一个守护进程脚本每 30 秒实现检测一次。

阶段 2: 如果同步出现如下错误号(1158,1159,1008,1007,1062, 则跳过错误。

阶段 3: 请使用数据技术实现上述脚本（获取主从判断及错误号部分）

模拟文本

```
[root@oldboy~]# mysql -uroot-p'oldboy' -S /data/3307/mysql.sock  
-e "show slavestatus\G;"
```

```
***** 1.row *****
```

```
Slave_IO_State:Waiting formaster to send event
```

```
Master_Host:10.0.0.179 #当前的 mysql master 服务器主机
```

```
Master_User: rep
```

```
Master_Port: 3306
```

```
Connect_Retry: 60
```

```
Master_Log_File:mysql-bin.000013
```

```
Read_Master_Log_Pos: 502547
```

```
Relay_Log_File:relay-bin.000013
```

```
Relay_Log_Pos:251
```

```
Relay_Master_Log_File:mysql-bin.000013
```

```
Slave_IO_Running:Yes
```

```
Slave_SQL_Running: Yes
```

```
Replicate_Do_DB:
```

Replicate_Ignore_DB: mysql

Replicate_Do_Table:

Replicate_Ignore_Table:

Replicate_Wild_Do_Table:

Replicate_Wild_Ignore_Table:

Last_Errno: 0

Last_Error:

Skip_Counter: 0

Exec_Master_Log_Pos: 502547

Relay_Log_Space:502986

Until_Condition:None

Until_Log_File:

Until_Log_Pos: 0

Master_SSL_Allowed: No

Master_SSL_CA_File:

Master_SSL_CA_Path:

Master_SSL_Cert:

Master_SSL_Cipher:

Master_SSL_Key:

Seconds_Behind_Master: 0 #和主库比同步延迟的秒数，这个参数很重要

Master_SSL_Verify_Server_Cert: No

Last_IO_Errno: 0

Last_IO_Error:

Last_SQL_Errno: 0

Last_SQL_Error:

脚本

```
[root@shell scripts]# vichcek_mysql_slave.sh
```

```
#!/bin/sh
```

```
#oldboy linux training
```

```
#2015-05-17
```

```
#说明：本脚本来自老男孩 linux21 期学员张耀开发！
```

```
# Source function library.
```

```
. /etc/init.d/functions
```

```
# Defined variables
```

```
MysqlUser=root
```

```
MysqlPass=oldboy123
```

```
MysqlPort=3307
```

```
Mysqlsock=/data/$MysqlPort/mysql.sock

ErrorNo=(1158 1159 1008 10071062)

errorlog=/tmp/error_skip.log

MysqlCmd="/application/mysql/bin/mysql-u$MysqlUser -p$MysqlPass
-S $Mysqlsock"

# Judge mysql server is ok?

[ -S $Mysqlsock ] ||{

    echo "Maybe MySQL have sometingwrong"

    exit 1

}

# Defined skip error Functions

function error_skip(){

    local flag

    flag=0

    for num in ${ErrorNo[@]}

    do

        if [ "$1" == "$num" ];then

            $MysqlCmd -e'stop slave;set global sql_slave_skip_coun
ter=1;start slave;'

            echo "$(date +%F_%R) $1">>$errorlog

        fi

    done

}
```

```

else

    echo "$(date +%F_%R) $1">>$errorlog

    ((flag++))

fi

done

[ "$flag" == "${#ErrorNo[@]}" ] &&{

    action "MySQL Slave"/bin/false

    uniq $errorlog|mail -s "MySQLSlave is error" 12345678@q
q.com

}

}

# Defined check slave Functions

function check_slave(){

    MyResult=`$MysqlCmd -e'show slavestatus\G'|egrep '_Running|
Behind_Master|SQL_Errno' |awk '{print $NF}'`

    array=($MyResult)

    if [ "${array[0]}" == "Yes" -a "${array[1]}" == "Yes" -a "${ar
ray[2]}" == "0" ]

    then

        action "MySQL Slave"/bin/true

```



```
    else

error_skip ${array[3]}

    fi
}

# Defined main Functions

function main(){

    while true

        do

            check_slave

            sleep 60

        done

    }

main
```

2、批量创建文件以及改名

企业面试题 2：使用 for 循环在 /oldboy 目录下通过随机小写 10 个字母，批量创建 10 个 html 文件，名称例如为：

```
[root@oldboy oldboy]# sh/server/scripts/oldboy.sh

[root@oldboy oldboy]# ls -l

total 0

-rw-r--r-- 1 root root 0 Apr 15 11:34 coaolvajcq_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 15 11:34 gmkhrancxh_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 jdxexendbe_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 qcawgsrtkp_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 qnvuxvicni_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 tmdjormaxr_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 ugaywanjlm_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 vfrphtqjpc_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 vioesjmcbu_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 wzewnojiwe_oldboy.html
-rw-r--r-- 1 root root 0 Apr 15 11:34 xzzruhdzda_oldboy.html
```

请使用至少两种方法实现，将以上文件名中的 `oldboy` 全部改成 `oldgirl`(用 `for` 循环实现), 并且 `html` 改成大写。

2.1 批量创建脚本

```
[root@shell scripts]# vitouch2.sh

#!/bin/bash

Path=/oldboy

[ -d $Path ] || mkdir $Path

for i in `seq 10`
do

    char=`echo $RANDOM|md5sum|cut -c 2-11|tr[0-9] [a-j]`

    touch $Path/${char}_oldboy.html
```

done

```
[root@shell scripts]# shtouch2.sh
```

```
[root@shell scripts]# ll -h/oldboy/
```

total 0

```
-rw-r--r-- 1 root root 0 Apr 1107:49 aeddbiacdj_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 bccbacghba_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 bdhijefefb_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 bhfghjcgaa_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 ddaacedijc_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 ebcbbfabaf_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 ecafccebbi_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 egcdafafad_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 iehfcgfaef_oldboy.html
```

```
-rw-r--r-- 1 root root 0 Apr 1107:49 ifddfaaicd_oldboy.html
```

2.2 批量改名

```
[root@shell scripts]# vimv2.sh
```

```
#!/bin/bash
```

```
Path=/oldboy
```

```
[ -d $Path ] && cd $Path
```

```
for file in `ls`  
  
do  
  
    mv $file `echo $file|sed -e"s#oldboy#oldgirl#g" -e "s#html#HTML#g" `  
  
done  
  
[root@shell scripts]# sh mv2.sh  
  
[root@shell scripts]# ll -h/oldboy/  
  
total 0  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 aeddbiacdj_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 bccbacghba_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 bdhijefefb_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 bhfghjcgaa_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 ddaacedijc_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 ebcbbfabaf_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 ecafccebbi_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 egcdafafad_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 iehfcgfaef_oldgirl.HTML  
  
-rw-r--r-- 1 root root 0 Apr 11 07:49 ifddfaaicd_oldgirl.HTML  
  
[root@shell scripts]# vimv3.sh
```

```
#!/bin/bash

Path=/oldboy

[ -d $Path ] && cd $Path

for file in `ls`

do

    mv $file `echo${file/oldboy.html/oldgirl.HTML}`

done

[root@shell scripts]# sh mv3.sh

[root@shell scripts]# ll -h/oldboy/

total 0

-rw-r--r-- 1 root root 0 Apr 1108:12 abcaeacdbe_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 affabgbccg_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 badbiffbfg_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 ccbcifibbe_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 cchbacgegb_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 cdfbjfdiib_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 chjechdgab_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 ghibcfabee_oldgirl.HTML
-rw-r--r-- 1 root root 0 Apr 1108:12 icafaafbdb_oldgirl.HTML
```

```
-rw-r--r-- 1 root root 0 Apr 11 08:12 igiijhbebj_oldgirl.HTML
```

3、批量创建用户随机密码

企业面试题 3：批量创建 10 个系统帐号 oldboy01-oldboy10 并设置密码（密码为随机 8 位字符串）。

```
[root@shell scripts]# viuseradd.sh

#!/bin/bash

. /etc/init.d/functions

Path=/server/scripts

UserDb=$Path/user.db

FailDb=$Path/fail_user.db

[ -d "$Path" ] || mkdir -p $Path

[ -f "$UserDb" ] || touch $UserDb

[ -f "$FailDb" ] || touch $FailDb

for n in $(seq -w 10)
do

    passwd=`echo $(date+%t%N)$RANDOM|md5sum|cut -c 2-9`

    useradd oldboy$n >&/dev/null&& user_status=$?

    echo "$passwd"|passwd --stdinoldboy$n >&/dev/null && pa
ss_status=$?

    if [ $user_status -eq 0 -a $pass_status -eq 0 ];then
```

```
        action "adduser oldboy$n"/bin/true

        echo -e "user:\toldboy$npass:$passwd" >>$UserDb

    else

        action "adduser oldboy$n"/bin/false

        echo -e "user:\toldboy$npass:$passwd" >>$FailDb

    fi

done

[root@shell scripts]# shuseradd.sh

adduser oldboy01

adduser oldboy02

adduser oldboy03

adduser oldboy04

adduser oldboy05

adduser oldboy06

adduser oldboy07

adduser oldboy08

adduser oldboy09

adduser oldboy10

[root@shell scripts]# cat user.db
```

```
user: oldboy01 pass:f3291720

user: oldboy02 pass:457a9f30

user: oldboy03 pass:ff186389

user: oldboy04 pass:8f884b7c

user: oldboy05 pass:26f831b4

user: oldboy06 pass:344e2300

user: oldboy07 pass:0736b278

user: oldboy08 pass:67c1fa76

user: oldboy09 pass:b11e7aa9

user: oldboy10 pass:9e0c3673
```

4、判断网络主机存活

企业面试题 4：写一个脚本，实现判断 10.0.0.0/24 网络里，当前在线用户的 IP 有哪些（方法有很多）

```
#!/bin/sh

for n in `seq 254`
do

    ping -c1 10.0.0.$n &>/dev/null

    if [ $? -eq 0 ]

    then
```



```
    echo "10.0.0.$n is up ">>/tmp/uplist.log

else

    echo "10.0.0.$n is down ">>/tmp/downlist.log

fi

done
```

5、解决 DOS 攻击生产案例

企业实战题 5: 请用至少两种方法实现! 写一个脚本解决 DOS 攻击生产案例。

提示: 根据 web 日志或者或者网络连接数, 监控当某个 IP 并发连接数或者短时间内 PV 达到 100, 即调用防火墙命令封掉对应的 IP, 监控频率每隔 3 分钟。防火墙命令为: iptables-AINPUT -s 10.0.1.10 -j DROP。

```
[root@shell scripts]# vi dos.sh

#!/bin/bash

log=/tmp/tmp.log

[ -f $log ] || touch $log

function add_iptables(){

    while read line

    do
```

```

        ip=`echo $line|awk '{print $2}'`

        count=`echo $line|awk '{print $1}'`

        if [ $count -gt 100 ] && [`iptables -L -n|grep "$ip"
|wc -l` -lt 1 ]

            then

                iptables -I INPUT -s $ip -jDROP

                echo "$line isdropped" >>/tmp/droplist.log

            fi

        done<$log
}

function main(){

    whiletrue

        do

            #awk '{print $1}' access.log|grep-v "^$" |sort|uniq
-c >$log

            netstat -an|grep EST|awk -F '[:]+' '{print $6}'|sor
t|uniq -c >$log

            add_iptables

```

```

        sleep 180

    done
}

main

```

6、MySQL 启动脚本

企业实战题 6：开发 mysql 多实例启动脚本：

已知 mysql 多实例启动命令为：

`mysqld_safe--defaults-file=/data/3306/my.cnf&`

停止命令为：`mysqladmin -u root -poldboy123 -S /data/3306/mysql.sock shutdown`

请完成 mysql 多实例启动脚本的编写

要求：用函数，case 语句、if 语句等实现。

```

[root@shell scripts]# vi mysqld1.sh

#!/bin/bash

# chkconfig: 2345 65 37

# description: manager multiplemysqld server.

# Warning: This script uses the/etc/init.d/functions system function.

#          System kernel version isCentOS6.6,2.6.32-504.el6.x86_64.

#          I'm not sure whether other systems can be used normal

```

y.

```
# Source function library.
```

```
. /etc/init.d/functions
```

```
Port=$2
```

```
datadir=/data/$2
```

```
Path="/application/mysql"
```

```
CmdPath="$Path/bin"
```

```
MysqlCmd="$CmdPath/mysqld_safe"
```

```
prog=mysqld
```

```
Port=$2
```

```
datadir=/data/$2
```

```
pidfile=${PIDFILE-$datadir/mysqld.pid}
```

```
lockfile=${LOCKFILE-/var/lock/subsys/mysqld$Port}
```

```
MysqlConf="$datadir/my.cnf"
```

```
RETVAL=0
```

```
Usage1(){
```

```
    echo -e "\033[33m USAGE: $0 {startport|stop port|status  
port|restart port} \033[0m"
```

```
    exit 6
```

```

}

Usage2(){

    echo -e "\033[33m MySQL DaemonProgram Need Configuration
File,like $datadir/my.cnf \033[0m"

    exit 6

}

[ -n "$Port" -a -z "`echo"${Port//[0-9]}/}"`" ] || Usage1

start(){

    [ -x $MysqlCmd ] || exit 5

    [ -f $MysqlConf ] || Usage2

    echo -n "Starting $prog: "

    daemon --pidfile=${pidfile}"$MysqlCmd --defaults-file=$M
ysqlConf &>/dev/null &"

    sleep 1

    RETVAL=$?

    echo

    [ $RETVAL = 0 ] && touch${lockfile}

    return $RETVAL

}

stop(){

```

```
    echo -n $"Stopping $prog: "  
  
    killproc -p $pidfile  
  
    retval=$?  
  
    echo  
  
    [ $retval -eq 0 ] && rm -f${lockfile}  
  
    return $retval  
}  
  
restart() {  
  
    stop  
  
    sleep 1  
  
    start  
}  
  
rh_status() {  
  
    status -p $pidfile  
}  
  
rh_status_q() {  
  
    rh_status >/dev/null 2>&1  
}  
  
case "$1" in
```

```
start)

    rh_status_q && exit 0

    $1

    ;;

stop)

    rh_status_q || exit 0

    $1

    ;;

restart)

    $1

    ;;

status)

    rh_status

    ;;

*)

    Usage1

    exit 2

esac
```

7、分库备份

企业实战题 7:如何实现对 MySQL 数据库进行分库备份，请用脚本实现

```
[root@shell scripts]# vifenku_backup.sh

#!/bin/bash

MysqlUser=root

PassWord=oldboy123

Port=3306

Socket="/data/$Port/mysql.sock"

MysqlCmd="mysql -u$MysqlUser-p$PassWord -S $Socket"

Database=`$MysqlCmd -e "showdatabases;"|egrep -v "Database|_sch
ema|mysql"`

Mysqldump="mysqldump-u$MysqlUser -p$PassWord -S $Socket"

IP=`ifconfig eth0|awk -F "[:]+" 'NR==2 {print $4}'`

BackupDir=/backup/$IP

[ -d $BackupDir ] || mkdir -p$BackupDir

for dbname in $Database

do

    $Mysqldump --events -B $dbname|gzip>/$BackupDir/${dbna
me}_${date +%F}_bak.sql.gz

done

[root@shell 10.0.0.3]# ls
```



```
chongtu_2015-04-11_bak.sql.gz  oldboy_2015-04-11_bak.sql.gz  test_2015-04-11_bak.sql.gz

mysql_2015-04-11_bak.sql.gz  qqqqqq_2015-04-11_bak.sql.gz
```

8、分库分表备份

企业实战题 8:如何实现对 MySQL 数据库进行分库加分表备份，请用脚本实现

```
[root@shell scripts]# vifenbiao_backup.sh

#!/bin/bash

MysqlUser=root

PassWord=oldboy123

Port=3306

Socket="/data/$Port/mysql.sock"

MysqlCmd="mysql -u$MysqlUser-p$PassWord -S $Socket"

Database=`$MysqlCmd -e "showdatabases;"|egrep -v "Database|_schema|mysql"`

Mysqldump="mysqldump-u$MysqlUser -p$PassWord -S $Socket"

IP=`ifconfig eth0|awk -F "[:]+" 'NR==2 {print $4}'`

BackupDir=/backup/$IP

[ -d $BackupDir ] || mkdir -p$BackupDir

for dbname in $Database

do
```

```

        [ ! -d /$BackupDir/$dbname ] &&mkdir -p /$BackupDir/$d
dbname

        TABLE=`$MysqlCmd -e "show tables from$dbname;"|sed '1d
        '

        for table in $TABLE

            do

                $Mysqldump $dbname $table|gzip>/$BackupDi
r/$dbname/${dbname}_${table}_${date +%F}.sql.gz

            done

        done
done

```

9、打印字母数不大于 6 的单词

企业面试题 9： **bash for** 循环打印下面这句话中字母数不大于 6 的单词(昆仑万维面试题)。

I am oldboy teacher welcome to oldboy training class.

请用至少两种方法实现！

```

[root@shell scripts]# vi length.sh

#!/bin/bash

echo "-----wc -L-----"

for word in I am oldboy teacherwelcome to oldboy training clas
s.

do

    if [ `echo ${word}|wc -L` -le 6 ]

```

```
        then

            echo $word

        fi

done

echo"-----{#word}-----"

for word in I am oldboy teacherwelcome to oldboy training clas
s.

do

    if [ ${#word} -le 6 ]

    then

        echo $word

    fi

done

echo "-----exprlength-----"

for word in I am oldboy teacherwelcome to oldboy training clas
s.

do

    if [ `expr length "$word"` -le 6]

    then

        echo $word
```

```
        fi

done

echo "-----{word:0:6}-----"

for word in I am oldboy teacherwelcome to oldboy training clas
s.

do

    if [ "$word" == "${word:0:6}" ]

    then

        echo $word

    fi

done

echo "-----awk-----"

echo "I am oldboy teacherwelcome to oldboy training class"|awk
'{for(i=1;i<=NF;i++)if(length($i)<=6)print $i}'

echo "-----awk2-----"

echo -n "I am oldboy teacherwelcome to oldboy training class."
| awk 'BEGIN {RS=FS} length($0)<=6{print $0}'

echo "-----数组-----"

arr=(I am oldboy teacher welcometo oldboy training class.)

for word in ${arr[@]}
```

```
do

    if [ ${#word} -le 6 ]

    then

        echo $word

    fi

done

echo-----

for((i=0;i<${#arr[*]};i++))

do

    if [ ${#arr[$i]} -le 6 ]

    then

        echo ${arr[$i]}

    fi

done

[root@shell scripts]# shlength.sh

-----wc -L-----

I

am

oldboy
```

to

oldboy

class.

-----{#word}-----

I

am

oldboy

to

oldboy

class.

-----expr length-----

I

am

oldboy

to

oldboy

class.

-----{word:0:6}-----

I

am

oldboy

to

oldboy

class.

-----awk-----

I

am

oldboy

to

oldboy

class

-----awk2-----

I

am

oldboy

to

oldboy

class.

-----数组-----

I

am

oldboy

to

oldboy

class.

I

am

oldboy

to

oldboy

class.

10、比较 2 个整数大小

11、打印选择菜单

12、监控 web、db 服务

12.1 监控 web 服务

12.1.1 本地端口判断

12.1.2 本地进程判断

12.1.3 远程端口判断

12.1.4 获取返回内容判断

12.1.5 根据状态码判断

12.2 监控 db 服务

12.2.1 插入与查询的值对比判断

12.2.2 返回值判断

12.2.3 本地端口对比

12.2.4 本地进程判断

12.2.5 远程端口判断

13、监控 memcache 服务

14、监控 web 站点目录

15、rsync 的系统启动脚本

16、抽奖

17、破解密码

18、批量检查多个网站地址是否正常

MySQL---Open 实例

1、MySQL 查询今天、昨天、近 7 天，近 30 天，本月，上一月的数据

假设有一张表 ad_proTrack_t 表，追踪产品时间字段为 crt_time

查询今天的信息记录

```
select * from ad_proTrack_t where to_days(crt_time) = to_days(now());
```

// 今天做测试的时候调用到这句 sql，发现不是想要的结果。

经过尝试发现，to_days 函数包括内的“时间字段”不能加引号，加引号的转换后为 NULL

查询昨天的信息记录

```
select * from ad_proTrack_t where to_days(now()) - to_days(crt_time) <= 1;
```

查询近 7 天的信息记录

```
select * from ad_proTrack_t where date_sub(curdate(), interval 7 day) <= date(crt_time);
```

查询近 30 天的信息记录

```
select * from ad_proTrack_t where date_sub(curdate(), interval 30 day) <= date(crt_time);
```

查询本月的信息记录

```
select * from ad_proTrack_t where  
date_format(crt_time, '%Y%m') = date_format(curdate(), '%Y%m');
```

查询上一月的信息记录:

```
select * from ad_proTrack_t where  
period_diff(date_format(now(), '%Y%m'), date_format(crt_time, '%Y%m')) = 1;
```

2、统计时间分布脚本

2.1、查询超时记录

```
select * from visit_record t where  
t.accesstime > to_date( '2013-09-1' , ' yyyy-MM-dd' ) and t.username is not null and  
t.result = ' 成功' and t.totaltimecost > 5000 and t.clientip <> ' 192.168.112.53' and  
t.clientip <> ' 192.168.112.200' and t.clientip <> ' 192.168.112.245' order by  
t.totaltimecost desc;
```

2.2、导出日期范围内的日志信息(已 xml 格式导出为 stat.xml,使用 navicat 工具导入进 stat 表)

```
select t.username,t.accesstime,t.totaltimecost,t.accessurl from visit_record t where t.accesstime>to_date('2013-09-1','yyyy-MM-dd') and t.username is not null and t.result='成功' and t.clientip <> '192.168.112.53' and t.clientip <> '192.168.112.200' and t.clientip <> '192.168.112.245' and (t.accessurl='/newOA/m_flow/navPage.do' or t.accessurl='/newOA/m_commWorkflow/submitNode_2p0.do' or t.accessurl='/newOA/user/showDaiBanxiangJsp.do' or t.accessurl='/newOA/user/showDaiBanXiangList.do' ) order by t.totaltimecost desc ;
```

2.3、统计超时率(mysql 库中)

```
select
    t.ACCESSURL as url
    ,COUNT(t.ACCESSURL) as 发生次数
    ,COUNT(IF(t.TOTALTIMECOST>5000,t.TOTALTIMECOST,null)) as 超时次数
    ,COUNT(IF(t.TOTALTIMECOST>5000,t.TOTALTIMECOST,null))/COUNT(t.ACCESSURL) as 超时率
    ,ROUND(avg(t.TOTALTIMECOST)) as 平均耗时
    ,max(t.TOTALTIMECOST) as 最大耗时
    from stat t
    group by t.ACCESSURL order by 平均耗时 desc;
```

2.4、时间分布(异常统计)

```
select
    t.ACCESSURL as url
    ,COUNT(IF(t.TOTALTIMECOST>5000 ,t.TOTALTIMECOST,null)) as 超时次数
    ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '09:00:00' and '10:00:00' ,t.TOTALTIMECOST,null)) as '9'
    ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '10:00:00' and '11:00:00' ,t.TOTALTIMECOST,null)) as '10'
```

```

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
1:00:00' and '12:00:00' ,t.TOTALTIMECOST,null)) as '11'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
2:00:00' and '13:00:00' ,t.TOTALTIMECOST,null)) as '12'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
3:00:00' and '14:00:00' ,t.TOTALTIMECOST,null)) as '13'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
4:00:00' and '15:00:00' ,t.TOTALTIMECOST,null)) as '14'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
5:00:00' and '16:00:00' ,t.TOTALTIMECOST,null)) as '15'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
6:00:00' and '17:00:00' ,t.TOTALTIMECOST,null)) as '16'

        ,COUNT(IF(t.TOTALTIMECOST>5000 and time(t.ACCESTIME) between '1
7:00:00' and '18:00:00' ,t.TOTALTIMECOST,null)) as '17'

        from stat t

        group by t.ACCESSURL with ROLLUP;

```

select

```

t.ACCESSURL as url

,COUNT(t.ACCESSURL) as 发生次数

,COUNT(IF( time(t.ACCESTIME) between '09:00:00' and '10:00:00'
,t.TOTALTIMECOST,null)) as '9'

,COUNT(IF( time(t.ACCESTIME) between '10:00:00' and '11:00:00'
,t.TOTALTIMECOST,null)) as '10'

,COUNT(IF( time(t.ACCESTIME) between '11:00:00' and '12:00:00'
,t.TOTALTIMECOST,null)) as '11'

,COUNT(IF( time(t.ACCESTIME) between '12:00:00' and '13:00:00'
,t.TOTALTIMECOST,null)) as '12'

,COUNT(IF( time(t.ACCESTIME) between '13:00:00' and '14:00:00'
,t.TOTALTIMECOST,null)) as '13'

,COUNT(IF( time(t.ACCESTIME) between '14:00:00' and '15:00:00'
,t.TOTALTIMECOST,null)) as '14'

,COUNT(IF( time(t.ACCESTIME) between '15:00:00' and '16:00:00'
,t.TOTALTIMECOST,null)) as '15'

```

```

        ,COUNT(IF( time(t.ACCESTIME) between '16:00:00' and '17:00:00'
,t.TOTALTIMECOST,null)) as '16'

        ,COUNT(IF( time(t.ACCESTIME) between '17:00:00' and '18:00:00'
,t.TOTALTIMECOST,null)) as '17'

from stat t

group by t.ACCESSURL with ROLLUP;

```

2.5、日期分布(正常统计)

```

select

    t.ACCESSURL as url

    ,COUNT(t.ACCESSURL) as 发生次数

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=0 ,t.ACCESTIME,null)) as '星
期 1'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=1 ,t.ACCESTIME,null)) as '星
期 2'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=2 ,t.ACCESTIME,null)) as '星
期 3'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=3 ,t.ACCESTIME,null)) as '星
期 4'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=4 ,t.ACCESTIME,null)) as '星
期 5'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=5 ,t.ACCESTIME,null)) as '星
期 6'

    ,COUNT(IF( WEEKDAY(t.ACCESTIME)=6 ,t.ACCESTIME,null)) as '日'

from stat t

group by t.ACCESSURL

```

```

select

    t.ACCESSURL as url

    ,COUNT(t.ACCESSURL) as 发生次数

```

```

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=0 ,t.
ACCESTIME,null)) as '星期 1'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=1 ,t.
ACCESTIME,null)) as '星期 2'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=2 ,t.
ACCESTIME,null)) as '星期 3'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=3 ,t.
ACCESTIME,null)) as '星期 4'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=4 ,t.
ACCESTIME,null)) as '星期 5'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=5 ,t.
ACCESTIME,null)) as '星期 6'

        ,COUNT(IF( t.TOTALTIMECOST>5000 and WEEKDAY(t.ACCESTIME)=6 ,t.
ACCESTIME,null)) as '日'

from stat t

group by t.ACCESSURL

```

3、连接 MySQL 数据库 PHP 代码

```

<?php
$host = "localhost";
$username = "database username";
$password = "database password";
$dbname = "database name";
$connection = mysql_connect($host, $username, $password)
Or die("Database Connection Failed");
$result = mysql_select_db($dbname)
Or die("database cannot be selected");
?>

```

4、Ruby 连接 mysql 代码

```

Require 'mysql'
M = mysql.new("localhost", "name", "password", "dbname");
R = m.query("select * from people order by name")
r.each_hash do |f|
  print "#{f['name']}-#{f['email']}"
end

```

5、mysql 批量删除

```
-- 首先创建存储过程
Create procedure del_line()
Begin
    Declare count_line INT;
    --先删除索引提高删除速度
    Alter table 'test_road_1'.ledp_logistics_line
    Drop index index_carried_id,
    Drop index index_state,
    Drop index index_leave_city,
    Drop index index_leave_city_area,
    Drop index index_arrive_city,
    Drop index index_arrive_city_area,
    Drop index index_mock_attachment;
    Select count(*) into count_line from leap_logistics_line;
    While count_line > 0 do
        Delete from ledp_logistics_line limit 10000;
        Commit;
        Set count_line = count_line - 10000;
    End while;
end
```

6、MySQL 中取得汉字字段的各汉字首字母

```
set global log_bin_trust_function_creators = 1;
```

```
set foreign_key_checks = 0;
```

```
-- -----
```

```
--Function structure for getPY
```

```
-- -----
```

```
DROP FUNCTION IF EXISTS getPY;
```

```
delimiter;;
```

```
create definer = 'root'@'%' function getPY(in_string varchar(65531)) returns mediumtext
charset utf8
```

```
begin
```

```
declare tmp_str varchar(65534) charset gbk default ""; # 截取字符串，每次做截取后的字符串存放在该变量中，初始为函数参数 in_string 值
```

```

declare tmp_len smallint default 0; # tmp_str 的长度

declare tmp_char varchar(2) charset gbk default ""; #截取字符，每次 left(tmp_str,1)返回值
存放在该变量中

declare tmp_rs varchar(65534) charset gbk default ""; #结果字符串

declare tmp_cc varchar(2) charset gbk default ""; #拼音字符，存放单个汉字对应的拼音首
字符

set tmp_str = in_string; #初始化,将 in_string 赋给 tmp_str

set tmp_len = length(tmp_str) # 初始化长度

while tmp_len>0 do #如果被计算的 tmp_str 长度大于 0 则进入该 while

set tmp_char = left(tmp_str,1); #获取 tmp_str 最左端的首个字符,注意这里是获取首个字
符，该字符可能是汉字，也可能不是。

set tmp_cc = tmp_char;#左端首个字符赋值给拼音字符

if length(tmp_char)>1 then # 判断左端首个字符是多字节还是单字节字符，要是多字节则
认为汉字且作以下拼音获取，要是单字节则不处理

select elt(interval
(conv(hex(tmp_char),16,10),0XB0A1,0XB0C5,0xB2C1,0XB4EE,0XB6EA,0xb7a2,0xb8a1,0xb9fe
,0xbbf7,0xbfa6,0xc0ac,0xc2eb,0xc4c3,0xc5b6,0xc5be,0xc6da,0xc8bb,0xc8f6,0xcbfa,0xcd
da,0xc
ef4,0xd1b9,0xd401),'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V',
'W','X','Y','Z') into tmp_cc; #获得汉字拼音首字符

end if;

set tmp_rs = concat(tmp_rs,tmp_cc);#将当前 tmp_str 左端首个字符拼音字符与返回字
符串拼接

set tmp_str = substring(tmp_str,2); #将 tmp_str 左端首字符去除

set tmp_len = length(tmp_str); # 计算当前字符串长度

end while;

return tmp_rs; # 返回结果字符串

end;;

delimiter;

```


7、MySQL 计算日期之间相差的天数

有两种方式可以获得 MySQL 两个日期之间的差值，一种是使用 `to_days` 函数，另一种是 `datediff` 函数

```
select id,to_days(now())-to_days(createtime) as dayFactor,datediff(now(),createtime) as  
dayFacotr1 from code_snippet limit 10
```

上面的 sql 中的 `code_snippet` 表中有字段 `id` 和 `createTime`。

9、MySQL 存储过程代码例子

--定义一个新的命令结束符号，默认的是以;为结束标记

-- 同样的可以通过 `delimiter;`在设置;为结束标记

```
delimiter $$
```

--删除函数 `rand_string`

```
drop function rand_string $$
```

--创建函数 `rand_string(n)`;随机产生 N 个字符组成的字符串

```
create function rand_string(n int)
```

```
returns varchar(255)
```

```
begin
```

```
declare chars_str varchar(100) default
```

```
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

```
declare return_str varchar(255) default '';
```

```
declare l int default 0;
```

```
while l<n do
```

```
--concat('a','b'):ab
```

```
--substring(str,pos,len):得到字符串 str 从 pos 位置开始长度为 len 的字符串
```

```
--rand(): 得到一个[0,1]的随机小数
```

```
set return_str = concat(return_str,substring(chars_str,floor(1+rand()*52),1));
```

```
set l = l+1;
```

end while;

return return_str;

end\$\$

delimiter;

10、 php 连接 mysql 的工具类

```
<?php
Class mysql {
    Private $defaultDB = null;
    Private $link = null;
    Private $sql = null;
    Private $bindValue = null;
    Public $num_rows = 0;
    Public $affected_rows = 0;
    Public $insert_id = 0;
    Public $queries = 0;
    Public function __construct() {
        If (func_num_args()) {
            $argv = func_get_arg(0);
            If (!empty($argv) && is_array($argv)) {
                $this->connect($argv);
                $argv['charset'] = isset($argv['charset']) ? $argv['charset'] : 'utf8';
                $this->setCharset($argv['charset']);
            }
        }

        Public function connect($argv, $charset = null) {
            If ($this->link) return false;
            $argv = func_get_arg(0);
            $argv['port'] = isset($argv['port']) ? $argv['port'] : 3306;
            $this->link = mysqli_connect($argv['host'], $argv['user'], $argv['password'],
            $argv['database'], $argv['port']);
            If (mysqli_connect_errno()) {
                Echo mysqli_connect_error(); exit(0);
            }
            $this->defaultDB = $argv['database'];

            If ($charset)
                $this->setCharset($charset);
        }
    }
}
```

```

public function selectDB($database){

    $int = mysqli_select_db($this->link, $database);

    if($int) $this->defaultDB = $database;

    return $int;

}

public function query($sql) {

    $result = mysqli_query($this->link, $sql);

    if(mysqli_errno($this->link)) { echo mysqli_error($this->link); exit(0); }

    $this->queries++;

    if(preg_match('/^use\\s+(\\w+)/', $sql, $matches)) list($range, $this->defaultDB) = $matches;

    $pattern = array('read'=> '/(?<=select|show)(.+)$/i', 'write'=> '/(?<=alter|use|replace|insert|update|delete)(.+)$/i');

    if(preg_match($pattern['write'], $sql)) {

        $this->affected_rows = mysqli_affected_rows($this->link);

    }else{

        $this->num_rows = mysqli_num_rows($result);

    }

    if(preg_match('/^insert(.+)$/i', $sql)) {

        $this->insert_id = mysqli_insert_id($this->link);

    }

```

```
        return $result;

    }

    public function find($sql) {

        $collection = array();

        $result = $this->query($sql);

        while($rows = mysqli_fetch_assoc($result))

            array_push($collection, $rows);

        mysqli_free_result($result);

        return $collection;

    }

    public function findOne($sql) {

        $result = $this->query($sql);

        $rows = mysqli_fetch_assoc($result);

        mysqli_free_result($result);

        return $rows;

    }

    public function setCharset($charset) {

        return mysqli_set_charset($this->link, $charset);

    }
```

```
public function prepare($sql) {

    $this->sql = $sql;

}

public function bindValue($search, $value) {

    $this->bindValue = array();

    $this->bindValue[$search] = $value;

}

public function execute() {

    if(func_num_args()) {

        $argv = func_get_arg(0);

        if(!empty($argv) && is_array($argv)) {

            if(!is_array($this->bindValue)) $this->bindValue = array();

            $this->bindValue = array_merge($this->bindValue, $argv);

        }

    }

    if($this->bindValue) {

        foreach($this->bindValue as $search => $value) {

            $this->sql = str_replace($search, $this->escape($value), $this->sql);

        }

        $this->bindValue = null;

    }

}
```

```
$int = $this->query($this->sql);

//$this->sql = null;

return (boolean) $int;

}


public function escape($string) {

    return mysqli_real_escape_string($this->link, $string);

}


public function close() {

    return mysqli_close($this->link);

}


public function ping() {

    return mysqli_ping($this->link);

}


public function autoCommit($boolean) {

    return mysqli_autocommit($this->link, $boolean);

}

}
```

```
public function commit() {

    return mysqli_commit($this->link);

}

public function rollback() {

    return mysqli_rollback($this->link);

}

public function __destruct() {

    if($this->link) $this->close();

    unset($this->link, $this->defaultDB, $this->bindValue, $this->sql, $this->result, $this->num_rows, $this->affected_rows, $this->insert_id);

}

}

$argv = array(
    'host' => 'localhost',
    'user' => 'root',
    'password' => '',
    'port' => 3306,
    'database' => 'test',
    'charset'=> 'utf8');
```

```
// Using the "mysql::__construct" method to connect MySQL database
```

```
$mysql = new mysql($argv);  
var_dump($mysql->find('select version()));  
var_dump($mysql->queries);
```

```
// Using the "mysql::connect" method to connect MySQL database
```

```
$mysql = new mysql();  
$mysql->connect($argv);  
var_dump($mysql->find('select version()));  
var_dump($mysql->queries);
```

```
$mysql = new mysql();  
$mysql->connect($argv);  
$mysql->setCharset($argv['charset']);  
var_dump($mysql->find('select version()));  
var_dump($mysql->queries);
```

```
?>
```

```
<?php
```

```
/*  
 * 名称：数据库连接类  
 * 介绍：适用于各种数据库链接  
 */
```

```
class mysql {
```



```

private $_link;

public function __construct($dbhost='localhost',$dbuser='root',$dbpassword='
',$dbname='taojindidai',$charset='gbk') {

    $this->_link = mysql_connect($dbhost,$dbuser,$dbpassword,true); /*连接数
数据库*/

    $this->_link or $this->errmsg('无法连接 MYSQL 服务器! '); /*是否连接成功*/

    if ($this->version() > '4.1') { /*检查数据库版本*/

        $this->query('set names '.$charset); /*设置数据库编码*/

    }

    /*打开数据库*/

    mysql_select_db($dbname,$this->_link) or $this->errmsg('无法连接数据库! ');

}

/*执行数据库操作*/

public function query($sql) {

    $result = mysql_query($sql,$this->_link);

    $result or $this->errmsg('执行 SQL 语句错误! ');

    return $result;

}

/*返回根据从结果集取得的行生成的数组*/

/*MYSQL_BOTH 得到一个同时包含关联和数字索引的数组 (如同 mysql_fetch_array())*/

/*MYSQL_ASSOC 得到一个同时包含关联和数字索引的数组 (如同 mysql_fetch_assoc())*/

/*MYSQL_NUM 得到一个同时包含关联和数字索引的数组 (如同 mysql_fetch_row())*/

public function fetch_array($result,$type = MYSQL_ASSOC) {

    return mysql_fetch_array($result,$type);

}

```

```
/*返回根据所取得的行生成的对象*/

public function fetch_object($result) {

    return mysql_fetch_object($result);

}


/*取得前一次 MySQL 操作所影响的记录行数*/

public function affected_rows() {

    return mysql_affected_rows($this->_link);

}


/* 释放结果内存*/

public function free_result($result) {

    return mysql_free_result($result);

}


/* 取得结果集中行的数目*/

public function num_rows($result) {

    return mysql_num_rows($result);

}


/* 取得结果集中字段的数目*/

public function num_fields($result) {

    return mysql_num_fields($result);

}


/*取得上一步 INSERT 操作产生的 ID*/

public function insert_id() {

    return mysql_insert_id($this->_link);

}
```

```

/* 发出 mysql 执行错误*/

private function errmsg($msg) {

    $message = '<strong>一个 MySQL 错误发生! </strong><br />';

    $message .= '<strong>错误号:</strong>'. mysql_errno($this->_link) . '<br />';

    $message .= '<strong>错误描述:</strong>'. $msg . mysql_error($this->_link) . '<br />';

    $message .= '<strong>错误时间:</strong>'. date('Y-m-d H:i:s');

    exit($message);

}

/*返回连接的标识*/

public function link_id() {

    return $this->_link;

}

/*返回数据库服务器版本*/

public function version() {

    return mysql_get_server_info($this->_link);

}

/*获得客户端真实的 IP 地址*/

function getip() {

    if(getenv("HTTP_CLIENT_IP") && strcasecmp(getenv("HTTP_CLIENT_IP"), "unknown")) {

        $ip = getenv("HTTP_CLIENT_IP");

    }elseif(getenv("HTTP_X_FORWARDED_FOR") && strcasecmp(getenv("HTTP_X_FORWARDED_FOR"), "unknown")) {

        $ip = getenv("HTTP_X_FORWARDED_FOR");

    }elseif(getenv("REMOTE_ADDR") && strcasecmp(getenv("REMOTE_ADDR"), "unknown")) {

```

```

        $ip = getenv("REMOTE_ADDR");

        }elseif(isset ($_SERVER['REMOTE_ADDR']) && $_SERVER['REMOTE_ADDR'] && strcmp($_SERVER['REMOTE_ADDR'], "unknown")) {
            $ip = "unknown";

        }

        $ip = $_SERVER['REMOTE_ADDR'];

    }else{

    }

    return ($ip);

}

}

?>

```

11、 MySQL 经纬度进行距离计算

公式如下,单位米:

第一点经纬度:lng1 lat1

第二点经纬度:lng2 lat2

```

round(6378.138*2*asin(sqrt(pow(sin( (lat1*pi()/180-lat2*pi()/180)/2),2)+cos(lat1*pi()/180)*cos(lat2*pi()/180)*

```

```

pow(sin( (lng1*pi()/180-lng2*pi()/180)/2),2)))*1000)

```

```

SELECT store_id,lng,lat,

```

```

        ROUND(6378.138*2*ASIN(SQRT(POW(SIN((22.299439*PI()/180-lat*PI()/180)/2),2)+COS(2
2.299439*PI()/180)*COS(lat*PI()/180)*POW(SIN((114.173881*PI()/180-lng*PI()/180)/2),
2)))*1000)

```

```

AS

```

```

    juli

```

```

FROM store_info having juli > 500

```

```

ORDER BY juli DESC

```

```

LIMIT 100

```

12、 MySQL 时间函数

FROM_UNIXTIME

FROM_UNIXTIME(unix_timestamp, format) 第一个参数是时间戳格式。第二个是最终想转换的格式，

```
SELECT FROM_UNIXTIME(1436102304,'%Y 年%m 月%d 日') as date;
```

结果 date: 2015 年 07 月 05 日

UNIX_TIMESTAMP

UNIX_TIMESTAMP(date) 则是将时间转化为时间戳

```
SELECT UNIX_TIMESTAMP('2015-07-05');
```

结果是:1436068800

示例：找出 2015-05 到 2015-07 log 表中的记录：

```
SELECT id, FROM_UNIXTIME(time,'%Y-%m-%d') as date
FROM log
WHERE time BETWEEN UNIX_TIMESTAMP('2015-05-01') AND UNIX_TIMESTAMP('2015-07-01');
```

13、 MySQL 游标使用模板

```
BEGIN
/**
 *
 * 游标模板
 * @author tsai
 * @email 2687547643@qq.com
 * @date 2017-01-03
 */
/* 游标数据变量 uid*/
Declare uid varchar(128);
Declare done int default 0;

/*查询所有用户*/
Declare cur cursor for select xxx from xx;
Declare continue handler for not found set done = 1;
Open cur;
```

```
/*开始循环*/
Repeat
/*提取游标里的数据*/
Fetch cur into uid;
  If done = 0 then
    执行你的业务
  End if;
Until done = 1
End repeat;
/*关闭游标*/
close cur;
End;
```

14、 MySQL 任务调度实现

今天有个业务需求,每天要重置流水号.想起 oracle 有 job 于是联想到 Mysql 应该有类似的.发现 mysql

通过 EVENT 来实现

语法如下

```
CREATE EVENT [IF NOT EXISTS] event_name
```

```
ON SCHEDULE schedule
```

```
[ON COMPLETION [NOT] PRESERVE]
```

```
[ENABLE | DISABLE]
```

```
[COMMENT 'comment']
```

```
DO sql_statement;
```

```
schedule:
```

```
AT TIMESTAMP [+ INTERVAL INTERVAL]
```

```
| EVERY INTERVAL [STARTS TIMESTAMP] [ENDS TIMESTAMP]
```

INTERVAL:

```
quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
```

```
WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
```

```
DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

简单使用如下

```
DELIMITER $$
```

```
/**
```

```
 * 重置流水号
```

```
 *
```

```
 * @author xuyw
```

```
 * @email xyw10000@163.com
```

```
 * @date 2014-05-06
```

```
 */
```

```
-- SET GLOBAL event_scheduler = ON$$      -- required for event to execute but not  
create
```

```
CREATE /*[DEFINER = { user | CURRENT_USER }]*/ EVENT `xxx`.`reset_serialNumber`
```

```
ON SCHEDULE EVERY 1 DAY STARTS '2014-05-06 23:59:59'
```

```
/* uncomment the example below you want to use */
```

```
-- scheduleexample 1: run once
```

```

        -- AT 'YYYY-MM-DD HH:MM.SS'/CURRENT_TIMESTAMP { + INTERVAL 1 [HOUR|MONTH|
WEEK|DAY|MINUTE|...] }

-- scheduleexample 2: run at intervals forever after creation

-- EVERY 1 [HOUR|MONTH|WEEK|DAY|MINUTE|...]

-- scheduleexample 3: specified start time, end time and interval for executi
on

/*EVERY 1 [HOUR|MONTH|WEEK|DAY|MINUTE|...]

STARTS CURRENT_TIMESTAMP/'YYYY-MM-DD HH:MM.SS' { + INTERVAL 1[HOUR|MONTH|W
EEK|DAY|MINUTE|...] }

ENDS CURRENT_TIMESTAMP/'YYYY-MM-DD HH:MM.SS' { + INTERVAL 1 [HOUR|MONTH|WE
EK|DAY|MINUTE|...] } */

/*[ON COMPLETION [NOT] PRESERVE]

[ENABLE | DISABLE]

[COMMENT 'comment']*/

DO

BEGIN

    UPDATE xxx_sequence

        SET current_value = 0

        WHERE id = 1;

END$$

DELIMITER ;

```


15、 php 调用 mysql 存储过程返回结果集

```
<?php

$db = new mysqli('localhost', 'root','', 'mydatabase');

$result = $db->query('CALL get_employees()');

while (list($employee_id, $name, $position) = $result->fetch_row()) {
    echo '$employeeid, $name, $position <br />';
}

?>
```

实用 SQL 语句大全

1、创建数据库

CREATE DATABASE database-name

2、删除数据库

DROP database dbname;

3、备份 sql server

4、 USE master

5、 EXEC sp_addumpdevice 'disk', 'testBack', 'c:\mssql7backup\MyNwind_1.dat'

6、 --- 开始 备份

7、 BACKUP DATABASE pubs TO testBack

4、创建新表

create table tabname(col1 type1 [not null] [primary key], col2 type2 [not null],...)

根据已有的表创建新表

A:create table tab_new like tab_old(使用旧表创建新表)

B: create table tab_new as select col1,col2,... from tab_old definition only

5、删除新表

drop table tabname

6、增加一个列

ALTER table tabname add column col type

注:列增加后将不能删除。DB2 中列加上后数据类型也不能改变,唯一能改变的是增加 varchar 类型的长度。

8、添加主键

ALTER table tabname add primary key(col)

删除主键: alter table tabname drop primary key(col)

9、创建索引

create [unique] index idxname on tabname(col...)

删除索引: drop index idxname

注:索引是不可更改的,想更改必须删除重新建

10、 创建视图

create view viewname as select statement

删除视图: drop view viewname

11、 几个简单的基本的 sql 语句

选择: select * from table1 where 范围

插入: insert into table(field1,filed2) values(value1,value2)

删除: delete from table1 where 范围

更新: update table1 set field1=value1 where 范围

查找: select * from table1 where field1 like ' %value1%' ---like 的语法很精妙,查资料!

排序: select * from table1 order by field1,field2 [desc]

总数: `select count as totalcount from table1`

求和: `select sum(field1) as sumvalue from table1`

平均: `select avg(field1) as avgvalue from table1`

最大: `select max(field1) as maxvalue from table1`

最小: `select min(field1) as minvalue from table1`

12、说明：几个高级查询运算词

A: UNION 运算符

UNION 运算符通过组合其他两个结果表(例如 TABLE1 和 TABLE2)并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时(即 UNION ALL)，不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

B: EXCEPT 运算符

EXCEPT 运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时 (EXCEPT ALL)，不消除重复行。

C: INTERSECT 运算符

INTERSECT 运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 ALL 随 INTERSECT 一起使用时 (INTERSECT ALL)，不消除重复行。

注：使用运算词的几个查询结果行必须是一致的。

13、说明：使用外连接

A、left (outer) join:

左外连接(左连接)：结果集几包括连接表的匹配行，也包括左连接表的所有行。

SQL: `select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c`

B: right (outer) join:

右外连接(右连接)：结果集既包括连接表的匹配连接行，也包括右连接表的所有行。

C: full/cross (outer) join:

全外连接：不仅包括符号连接表的匹配行，还包括两个连接表中的所有记录。

14、分组:Group by:

一张表，一旦分组 完成后，查询后只能得到组相关的信息。

组相关的信息：（统计信息）count, sum, max, min, avg 分组的标准）

在 SQLServer 中分组时：不能以 text, ntext, image 类型的字段作为分组依据

在 selecte 统计函数中的字段，不能和普通的字段放在一起；

15、对数据库进行操作：

分离数据库： sp_detach_db；附加数据库：sp_attach_db 后接表明，附加需要完整的路径名

16、如何修改数据库的名称：

```
sp_renamedb 'old_name', 'new_name'
```

二、提升

1、说明：复制表(只复制结构,源表名：a 新表名：b) (Access 可用)

法一：select * into b from a where 1<>1(仅用于 SQLServer)

法二：select top 0 * into b from a

2、说明：拷贝表(拷贝数据,源表名：a 目标表名：b) (Access 可用)

```
insert into b(a, b, c) select d,e,f from b;
```

3、说明：跨数据库之间表的拷贝(具体数据使用绝对路径) (Access 可用)

```
insert into b(a, b, c) select d,e,f from b in '具体数据库' where 条件
```

例子：..from b in '"&Server.MapPath(".")&"\data.mdb" &"' where..

4、说明：子查询(表名 1: a 表名 2: b)

```
select a,b,c from a where a IN (select d from b ) 或者: select a,b,c from a where  
a IN (1,2,3)
```

5、说明：显示文章、提交人和最后回复时间

```
select a.title,a.username,b.adddate from table a, (select max(adddate) adddate  
from table where table.title=a.title) b
```

6、说明：外连接查询(表名 1: a 表名 2: b)

```
select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c
```

7、说明：在线视图查询(表名 1: a)

```
select * from (SELECT a,b,c FROM a) T where t.a > 1;
```

8、说明：**between** 的用法,between 限制查询数据范围时包括了边界值,not between 不包括

```
select * from table1 where time between time1 and time2
```

```
select a,b,c, from table1 where a not between 数值1 and 数值2
```

9、说明：in 的使用方法

```
select * from table1 where a [not] in ( '值1' , '值2' , '值4' , '值6' )
```

10、说明：两张关联表，删除主表中已经在副表中没有的信息

```
delete from table1 where not exists ( select * from table2 where  
table1.field1=table2.field1 )
```

11、说明：四表联查问题：

```
select * from a left inner join b on a.a=b.b right inner join c on a.a=c.c inner  
join d on a.a=d.d where .....
```

12、说明：日程安排提前五分钟提醒

```
SQL: select * from 日程安排 where datediff('minute',f开始时间,getdate())>5
```

13、说明：一条 sql 语句搞定数据库分页

```
select top 10 b.* from (select top 20 主键字段,排序字段 from 表名 order by 排序字段 desc) a,表名 b where b.主键字段 = a.主键字段 order by a.排序字段
```

具体实现：

关于数据库分页：

```
declare @start int,@end int
```

```
@sql nvarchar(600)
```

```
set @sql=' select top' +str(@end-@start+1)+' +from T where rid not in(select top' +str(@str-1)+' Rid from T where Rid>-1)'
```

```
exec sp_executesql @sql
```

注意：在 top 后不能直接跟一个变量，所以在实际应用中只有这样的进行特殊的处理。Rid 为一个标识列，如果 top 后还有具体的字段，这样做是非常有好处的。因为这样可以避免 top 的字段如果是逻辑索引的，查询的结果后实际表中的不一致(逻辑索引中的数据有可能和数据表中的不一致，而查询时如果处在索引则首先查询索引)

14、说明：前 10 条记录

```
select top 10 * form table1 where 范围
```

15、说明：选择在每一组 **b** 值相同的数据中对应的 **a** 最大的记录的所有信息(类似这样的用法可以用于论坛每月排行榜,每月热销产品分析,按科目成绩排名,等等.)

```
select a,b,c from tablename ta where a=(select max(a) from tablename tb where tb.b=ta.b)
```

16、说明：包括所有在 **TableA** 中但不在 **TableB** 和 **TableC** 中的行并消除所有重复行而派生出一个结果表

```
(select a from tableA ) except (select a from tableB) except (select a from tableC)
```

17、说明：随机取出 10 条数据

```
select top 10 * from tablename order by newid()
```

18、说明：随机选择记录

```
select newid()
```

19、说明：删除重复记录

```
1),delete from tablename where id not in (select max(id) from tablename group by col1,col2,...)
```

```
2),select distinct * into temp from tablename
```

```
delete from tablename
```

```
insert into tablename select * from temp
```

评价： 这种操作牵连大量的数据的移动，这种做法不适合大容量但数据操作

3), 例如： 在一个外部表中导入数据，由于某些原因第一次只导入了一部分，但很难判断具体位置，这样只有在下一次全部导入，这样也就产生好多重复的字段，怎样删除重复字段

```
alter table tablename
```

--添加一个自增列

```
add column_b int identity(1,1)
```

```
delete from tablename where column_b not in(
```

```
select max(column_b) from tablename group by column1,column2,...)
```

```
alter table tablename drop column column_b
```

20、说明：列出数据库里所有的表名

```
select name from sysobjects where type='U' // U代表用户
```

21、说明：列出表里的所有的列名

```
select name from syscolumns where id=object_id('TableName')
```

22、说明：列示 type、vender、pcs 字段，以 type 字段排列，case 可以方便地实现多重选择，类似 select 中的 case。

```
select type, sum(case vender when 'A' then pcs else 0 end), sum(case vender when 'C' then pcs else 0 end), sum(case vender when 'B' then pcs else 0 end) FROM tablename group by type
```

显示结果：

type	vender	pcs
------	--------	-----

电脑	A	1
----	---	---

电脑	A	1
----	---	---

光盘	B	2
----	---	---

光盘	A	2
----	---	---

手机	B	3
----	---	---

手机	C	3
----	---	---

23、说明：初始化表 table1

```
TRUNCATE TABLE table1
```

24、说明：选择从 10 到 15 的记录

```
select top 5 * from (select top 15 * from table order by id asc) table_别名 order by id desc
```

三、技巧

1、1=1, 1=2 的使用，在 SQL 语句组合时用的较多

“where 1=1” 是表示选择全部 “where 1=2” 全部不选，

如:

```
if @strWhere !=''  
  
begin  
  
set @strSQL='select count(*) as Total from [' + @tblName + '] where ' + @strWhere  
  
end  
  
else  
  
begin  
  
set @strSQL='select count(*) as Total from [' + @tblName + ']'  
  
end
```

我们可以直接写成

错误!未找到目录项。

```
set @strSQL='select count(*) as Total from [' + @tblName + '] where 1=1 安  
定 ' + @strWhere
```

2、收缩数据库

--重建索引

DBCC REINDEX

DBCC INDEXDEFRAG

--收缩数据和日志

DBCC SHRINKDB

DBCC SHRINKFILE

3、压缩数据库

```
dbcc shrinkdatabase(dbname)
```

4、转移数据库给新用户以已存在用户权限

```
exec sp_change_users_login 'update_one','newname','oldname'  
  
go
```

5、检查备份集

```
RESTORE VERIFYONLY from disk='E:\dvbbs.bak'
```

6、修复数据库

```
ALTER DATABASE [dvbbs] SET SINGLE_USER  
  
GO  
  
DBCC CHECKDB('dvbbs',repair_allow_data_loss) WITH TABLOCK  
  
GO  
  
ALTER DATABASE [dvbbs] SET MULTI_USER  
  
GO
```

7、日志清除

```
SET NOCOUNT ON  
  
DECLARE @LogicalFileName sysname,  
  
@MaxMinutes INT,  
  
@NewSize INT  
  
USE tablename -- 要操作的数据库名  
  
SELECT @LogicalFileName = 'tablename_log', -- 日志文件名  
  
@MaxMinutes = 10, -- Limit on time allowed to wrap log.  
  
@NewSize = 1 -- 你想设定的日志文件的大小(M)  
  
Setup / initialize
```

```

DECLARE @OriginalSize int

SELECT @OriginalSize = size

FROM sysfiles

WHERE name = @LogicalFileName

SELECT 'Original Size of ' + db_name() + ' LOG is ' +

CONVERT(VARCHAR(30),@OriginalSize) + ' 8K pages or ' +

CONVERT(VARCHAR(30), (@OriginalSize*8/1024)) + ' MB'

FROM sysfiles

WHERE name = @LogicalFileName

CREATE TABLE DummyTrans

(DummyColumn char (8000) not null)

DECLARE @Counter INT,

@StartTime DATETIME,

@TruncLog VARCHAR(255)

SELECT @StartTime = GETDATE(),

@TruncLog = 'BACKUP LOG ' + db_name() + ' WITH TRUNCATE_ONLY'

DBCC SHRINKFILE (@LogicalFileName, @NewSize)

EXEC (@TruncLog)

-- Wrap the log if necessary.

WHILE @MaxMinutes > DATEDIFF (mi, @StartTime, GETDATE()) -- time has not expired

AND @OriginalSize = (SELECT size FROM sysfiles WHERE name = @LogicalFileName)

AND (@OriginalSize * 8 /1024) > @NewSize

```

```

BEGIN -- Outer loop.

SELECT @Counter = 0

WHILE ((@Counter < @OriginalSize / 16) AND (@Counter < 50000))

BEGIN -- update

INSERT DummyTrans VALUES (' Fill Log') DELETE DummyTrans

SELECT @Counter = @Counter + 1

END

EXEC (@TruncLog)

END

SELECT 'Final Size of ' + db_name() + ' LOG is ' +

CONVERT(VARCHAR(30),size) + ' 8K pages or ' +

CONVERT(VARCHAR(30), (size*8/1024)) + 'MB'

FROM sysfiles

WHERE name = @LogicalFileName

DROP TABLE DummyTrans

SET NOCOUNT OFF

```

8、说明：更改某个表

```
exec sp_changeobjectowner 'tablename','dbo'
```

9、存储更改全部表

```

CREATE PROCEDURE dbo.User_ChangeObjectOwnerBatch

@OldOwner as NVARCHAR(128),

@NewOwner as NVARCHAR(128)

```

```

AS

DECLARE @Name as NVARCHAR(128)

DECLARE @Owner as NVARCHAR(128)

DECLARE @OwnerName as NVARCHAR(128)

DECLARE curObject CURSOR FOR

select 'Name' = name,

'Owner' = user_name(uid)

from sysobjects

where user_name(uid)=@OldOwner

order by name

OPEN curObject

FETCH NEXT FROM curObject INTO @Name, @Owner

WHILE (@@FETCH_STATUS=0)

BEGIN

if @Owner=@OldOwner

begin

set @OwnerName = @OldOwner + '.' + rtrim(@Name)

exec sp_changeobjectowner @OwnerName, @NewOwner

end

-- select @name, @NewOwner, @OldOwner

FETCH NEXT FROM curObject INTO @Name, @Owner

END

```

```
close curObject

deallocate curObject

GO
```

10、SQL SERVER 中直接循环写入数据

```
declare @i int

set @i=1

while @i<30

begin

insert into test (userid) values(@i)

set @i=@i+1

end
```

案例：

有如下表，要求就表中所有沒有及格的成績，在每次增長 0.1 的基礎上，使他們剛好及格：

Name	score
------	-------

Zhangshan	80
-----------	----

Lishi	59
-------	----

Wangwu	50
--------	----

Songquan	69
----------	----

```
while((select min(score) from tb_table)<60)
```

```
begin
```

```
update tb_table set score =score*1.01
```

```
where score<60

if (select min(score) from tb_table)>60

break

else

continue

end
```

数据开发-经典

1.按姓氏笔画排序:

```
Select * From TableName Order By CustomerName Collate Chinese_PRC_Stroke_ci_as
//从少到多
```

2.数据库加密:

```
select encrypt('原始密码')

select pwdencrypt('原始密码')

select pwdcompare('原始密码','加密后密码') = 1--相同;否则不相同 encrypt('原始密码')

select pwdencrypt('原始密码')

select pwdcompare('原始密码','加密后密码') = 1--相同;否则不相同
```

3.取回表中字段:

```
declare @list varchar(1000),

@sql nvarchar(1000)

select @list=@list+', '+b.name from sysobjects a, syscolumns b where a.id=b.id
and a.name='表 A'
```

```
set @sql='select '+right(@list,len(@list)-1)+' from 表 A'

exec (@sql)
```

4.查看硬盘分区:

```
EXEC master..xp_fixeddrives
```

5.比较 A,B 表是否相等:

```
if (select checksum_agg(binary_checksum(*)) from A)

=

(select checksum_agg(binary_checksum(*)) from B)

print '相等'

else

print '不相等'
```

6.杀掉所有的事件探查器进程:

```
DECLARE hcforeach CURSOR GLOBAL FOR SELECT 'kill '+RTRIM(spId) FROM
master.dbo.sysprocesses

WHERE program_name IN('SQL profiler',N'SQL 事件探查器')

EXEC sp_msforeach_worker '?'
```

7.记录搜索:

开头到 N 条记录

```
Select Top N * From 表
```

N 到 M 条记录(要有主索引 ID)

```
Select Top M-N * From 表 Where ID in (Select Top M ID From 表) Order by ID Desc
```

N 到结尾记录

Select Top N * From 表 Order by ID Desc

案例

例如 1: 一张表有一万多条记录, 表的第一个字段 RecID 是自增长字段, 写一个 SQL 语句, 找出表的第 31 到第 40 个记录。

```
select top 10 recid from A where recid not in(select top 30 recid from A)
```

分析: 如果这样写会产生某些问题, 如果 recid 在表中存在逻辑索引。

select top 10 recid from A where.....是从索引中查找, 而后面的 select top 30 recid from A 则在数据表中查找, 这样由于索引中的顺序有可能和数据表中的不一致, 这样就导致查询到的不是本来的欲得到的数据。

解决方案

1, 用 order by select top 30 recid from A order by ricid 如果该字段不是自增长, 就会出现问題

2, 在那个子查询中也加条件: select top 30 recid from A where recid>-1

例 2: 查询表中的最后以条记录, 并不知道这个表共有多少数据, 以及表结构。

```
set @s = 'select top 1 * from T where pid not in (select top ' + str(@count-1) + ' pid from T)'
```

```
print @s exec sp_executesql @s
```

9: 获取当前数据库中的所有用户表

```
select Name from sysobjects where xtype='u' and status>=0
```

10: 获取某一个表的所有字段

```
select name from syscolumns where id=object_id('表名')
```

```
select name from syscolumns where id in (select id from sysobjects where type = 'u' and name = '表名')
```

两种方式的效果相同

11: 查看与某一个表相关的视图、存储过程、函数

```
select a.* from sysobjects a, syscomments b where a.id = b.id and b.text like  
'%表名%'
```

12: 查看当前数据库中所有存储过程

```
select name as 存储过程名称 from sysobjects where xtype='P'
```

13: 查询用户创建的所有数据库

```
select * from master..sysdatabases D where sid not in(select sid from  
master..syslogins where name='sa')
```

或者

```
select dbid, name AS DB_NAME from master..sysdatabases where sid <> 0x01
```

14: 查询某一个表的字段和数据类型

```
select column_name,data_type from information_schema.columns  
where table_name = '表名'
```

15: 不同服务器数据库之间的数据操作

--创建链接服务器

```
exec sp_addlinkedserver 'ITSV ', ' ', 'SQLOLEDB ', '远程服务器名或 ip 地址 '
```

```
exec sp_addlinkedsrvlogin 'ITSV ', 'false ',null, '用户名 ', '密码 '
```

--查询示例

```
select * from ITSV.数据库名.dbo.表名
```

--导入示例

```
select * into 表 from ITSV.数据库名.dbo.表名
```

--以后不再使用时删除链接服务器

```

exec sp_dropserver 'ITSV ', 'droplogins '

--连接远程/局域网数据(openrowset/openquery/opendatasource)

--1、openrowset

--查询示例

select * from openrowset( 'SQLOLEDB ', 'sql 服务器名 '; '用户名 '; '密码 ',
数据库名.dbo.表名)

--生成本地表

select * into 表 from openrowset( 'SQLOLEDB ', 'sql 服务器名 '; '用户名 '; '
密码 ',数据库名.dbo.表名)

--把本地表导入远程表

insert openrowset( 'SQLOLEDB ', 'sql 服务器名 '; '用户名 '; '密码 ',数据库
名.dbo.表名)

select *from 本地表

--更新本地表

update b

set b.列A=a.列A

from openrowset( 'SQLOLEDB ', 'sql 服务器名 '; '用户名 '; '密码 ',数据库名.dbo.
表名)as a inner join 本地表 b

on a.column1=b.column1

--openquery 用法需要创建一个连接

--首先创建一个连接创建链接服务器

exec sp_addlinkedserver 'ITSV ', ' ', 'SQLOLEDB ', '远程服务器名或 ip 地址 '

--查询

select *

```

```
FROM openquery(ITSV, 'SELECT * FROM 数据库.dbo.表名')
```

--把本地表导入远程表

```
insert openquery(ITSV, 'SELECT * FROM 数据库.dbo.表名')
```

```
select * from 本地表
```

--更新本地表

```
update b
```

```
set b.列B=a.列B
```

```
FROM openquery(ITSV, 'SELECT * FROM 数据库.dbo.表名') as a
```

```
inner join 本地表 b on a.列A=b.列A
```

--3、opendatasource/openrowset

```
SELECT *
```

```
FROM opendatasource('SQLOLEDB', 'Data Source=ip/ServerName;User ID=登陆名;Password=密码').test.dbo.roy_ta
```

--把本地表导入远程表

```
insert opendatasource('SQLOLEDB', 'Data Source=ip/ServerName;User ID=登陆名;Password=密码').数据库.dbo.表名
```

```
select * from 本地表
```

SQL Server 基本函数

SQL Server 基本函数

1. 字符串函数 长度与分析用

1, datalength(Char_expr) 返回字符串包含字符数, 但不包含后面的空格

2, substring(expression, start, length) 取子串, 字符串的下标是从“1”, start 为起始位置, length 为字符串长度, 实际应用中以 len(expression) 取得其长度

3, right(char_expr, int_expr) 返回字符串右边第 int_expr 个字符, 还用 left 于之相反

4, isnull(check_expression , replacement_value) 如果 check_expression 為空, 則返回 replacement_value 的值, 不為空, 就返回 check_expression 字符操作类

5, Sp_addtype 自定義數據類型

例如: EXEC sp_addtype birthday, datetime, 'NULL'

6, set nocount {on|off}

使返回的结果中不包含有关受 Transact-SQL 语句影响的行数的信息。如果存储过程中包含的一些语句并不返回许多实际的数据, 则该设置由于大量减少了网络流量, 因此可显著提高性能。SET NOCOUNT 设置是在执行或运行时设置, 而不是在分析时设置。

SET NOCOUNT 为 ON 时, 不返回计数(表示受 Transact-SQL 语句影响的行数)。

SET NOCOUNT 为 OFF 时, 返回计数

常识

在 SQL 查询中: from 后最多可以跟多少张表或视图: 256

在 SQL 语句中出现 Order by, 查询时, 先排序, 后取

在 SQL 中, 一个字段的最大容量是 8000, 而对于 nvarchar(4000), 由于 nvarchar 是 Unicode 码。

SQLServer2000 同步复制技术实现步骤

一、 预备工作

1. 发布服务器, 订阅服务器都创建一个同名的 windows 用户, 并设置相同的密码, 做为发布快照文件夹的有效访问用户

--管理工具

--计算机管理

--用户和组

--右键用户

--新建用户

--建立一个隶属于 administrator 组的登陆 windows 的用户 (SynUser)

2. 在发布服务器上, 新建一个共享目录, 做为发布的快照文件的存放目录, 操作:

我的电脑--D:\ 新建一个目录, 名为: PUB

--右键这个新建的目录

--属性--共享

--选择"共享该文件夹"

--通过"权限"按钮来设置具体的用户权限, 保证第一步中创建的用户 (SynUser) 具有对该文件夹的所有权限

--确定

3. 设置 SQL 代理 (SQLSERVERAGENT) 服务的启动用户 (发布/订阅服务器均做此设置)

开始--程序--管理工具--服务

--右键 SQLSERVERAGENT

--属性--登陆--选择"此账户"

--输入或者选择第一步中创建的 windows 登录用户名 (SynUser)

--"密码"中输入该用户的密码

4. 设置 SQL Server 身份验证模式, 解决连接时的权限问题 (发布/订阅服务器均做此设置)

企业管理器

--右键 SQL 实例--属性

--安全性--身份验证

--选择"SQL Server 和 Windows"

--确定

5. 在发布服务器和订阅服务器上互相注册

企业管理器

--右键 SQL Server 组

--新建 SQL Server 注册...

--下一步--可用的服务器中, 输入你要注册的远程服务器名 --添加

--下一步--连接使用, 选择第二个“SQL Server 身份验证”

--下一步--输入用户名和密码(SynUser)

--下一步--选择 SQL Server 组, 也可以创建一个新组

--下一步--完成

6. 对于只能用 IP, 不能用计算机名的, 为其注册服务器别名(此步在实施中没用到)

(在连接端配置, 比如, 在订阅服务器上配置的话, 服务器名称中输入的是发布服务器的 IP)

开始--程序--Microsoft SQL Server--客户端网络实用工具

--别名--添加

--网络库选择“tcp/ip”--服务器别名输入 SQL 服务器名

--连接参数--服务器名称中输入 SQL 服务器 ip 地址

--如果你修改了 SQL 的端口, 取消选择“动态决定端口”, 并输入对应的端口号

二、 正式配置

1、配置发布服务器

打开企业管理器, 在发布服务器(B、C、D)上执行以下步骤:

(1) 从[工具]下拉菜单的[复制]子菜单中选择[配置发布、订阅服务器和分发]出现配置发布和分发向导

(2) [下一步] 选择分发服务器 可以选择把发布服务器自己作为分发服务器或者其他 sql 的服务器(选择自己)

(3) [下一步] 设置快照文件夹

采用默认\\servername\Pub

(4) [下一步] 自定义配置

可以选择:是, 让我设置分发数据库属性启用发布服务器或设置发布设置

否, 使用下列默认设置(推荐)

(5) [下一步] 设置分发数据库名称和位置 采用默认值

(6) [下一步] 启用发布服务器 选择作为发布的服务器

(7) [下一步] 选择需要发布的数据库和发布类型

(8) [下一步] 选择注册订阅服务器

(9) [下一步] 完成配置

2、创建出版物

发布服务器 B、C、D 上

(1) 从[工具]菜单的[复制]子菜单中选择[创建和管理发布]命令

(2) 选择要创建出版物的数据库，然后单击[创建发布]

(3) 在[创建发布向导]的提示对话框中单击[下一步]系统就会弹出一个对话框。对话框上的内容是复制的三个类型。我们现在选第一个也就是默认的快照发布(其他两个大家可以去看看帮助)

(4) 单击[下一步]系统要求指定可以订阅该发布的数据库服务器类型，

SQLSERVER 允许在不同的数据库如 orACLE 或 ACCESS 之间进行数据复制。

但是在这里我们选择运行“SQL SERVER 2000”的数据库服务器

(5) 单击[下一步]系统就弹出一个定义文章的对话框也就是选择要出版的表

注意：如果前面选择了事务发布 则再这一步中只能选择带有主键的表

(6) 选择发布名称和描述

(7) 自定义发布属性 向导提供的选择：

是 我将自定义数据筛选, 启用匿名订阅和或其他自定义属性

否 根据指定方式创建发布 (建议采用自定义的方式)

(8) [下一步] 选择筛选发布的方式

(9) [下一步] 可以选择是否允许匿名订阅

1) 如果选择署名订阅, 则需要在发布服务器上添加订阅服务器

方法: [工具]->[复制]->[配置发布、订阅服务器和分发的属性]->[订阅服务器] 中添加

否则在订阅服务器上请求订阅时会出现的提示: 改发布不允许匿名订阅

如果仍然需要匿名订阅则用以下解决办法

[企业管理器]->[复制]->[发布内容]->[属性]->[订阅选项] 选择允许匿名请求订阅

2) 如果选择匿名订阅, 则配置订阅服务器时不会出现以上提示

(10) [下一步] 设置快照 代理程序调度

(11) [下一步] 完成配置

当完成出版物的创建后创建出版物的数据库也就变成了一个共享数据库

有数据

srv1. 库名..author 有字段: id, name, phone,

srv2. 库名..author 有字段: id, name, telephone, adress

要求:

srv1. 库名..author 增加记录则 srv1. 库名..author 记录增加

srv1.库名..author 的 phone 字段更新, 则 srv1.库名..author 对应字段 telephone 更新

--*/

--大致的处理步骤

--1. 在 srv1 上创建连接服务器, 以便在 srv1 中操作 srv2, 实现同步

exec sp_addlinkedserver 'srv2','','SQLLEDB','srv2 的 sql 实例名或 ip'

exec sp_addlinkedsrvlogin 'srv2','false',null,'用户名','密码'

go

--2. 在 srv1 和 srv2 这两台电脑中, 启动 msdtc(分布式事务处理服务), 并且设置为自动启动

。我的电脑--控制面板--管理工具--服务--右键 Distributed Transaction Coordinator--属性--启动--并将启动类型设置为自动启动

go

--然后创建一个作业定时调用上面的同步处理存储过程就行了

企业管理器

--管理

--SQL Server 代理

--右键作业

--新建作业

--"常规"项中输入作业名称

--"步骤"项

--新建

--"步骤名"中输入步骤名

--"类型"中选择"Transact-SQL 脚本(TSQL)"

--"数据库"选择执行命令的数据库

--"命令"中输入要执行的语句: exec p_process

--确定

--"调度"项

--新建调度

--"名称"中输入调度名称

--"调度类型"中选择你的作业执行安排

--如果选择"反复出现"

--点"更改"来设置你的时间安排

然后将 SQL Agent 服务启动, 并设置为自动启动, 否则你的作业不会被执行

设置方法:

我的电脑--控制面板--管理工具--服务--右键 SQLSERVERAGENT--属性--启动类型--选择"自动启动"--确定.

--3. 实现同步处理的方法 2, 定时同步

--在 srv1 中创建如下的同步处理存储过程

```
create proc p_process
```

```
as
```

--更新修改过的数据

```
update b set name=i.name,telephone=i.telephone
```

```
from srv2.库名.dbo.author b,author i
```

```
where b.id=i.id and
```

```
(b.name <> i.name or b.telphone <> i.telphone)
```

--插入新增的数据

```
insert srv2.库名.dbo.author(id,name,telphone)
```

```
select id,name,telphone from author i
```

```
where not exists(
```

```
select * from srv2.库名.dbo.author where id=i.id)
```

--删除已经删除的数据(如果需要的话)

```
delete b
```

```
from srv2.库名.dbo.author b
```

```
where not exists(
```

```
select * from author where id=b.id)
```

```
go
```