数据库的高级语法

建立表

```
#truncate  #把表的数据全部清楚,并且id从1开始

CREATE TABLE `stuinfo` (
  `sid` int(11) NOT NULL AUTO_INCREMENT,
  `sname` varchar(255) DEFAULT NULL,
  `sex` enum('男','女') DEFAULT NULL,
  `age` tinyint(4) DEFAULT NULL,
  `city` varchar(64) DEFAULT NULL,
  `seat` tinyint(2) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB AUTO_INCREMENT=29 DEFAULT
CHARSET=utf8mb4;


CREATE TABLE `score` (
  `sid` int(11) NOT NULL COMMENT '学号(主
键)',
  `ch` tinyint(4) DEFAULT NULL COMMENT '语文
成绩',
  `math` tinyint(4) DEFAULT NULL COMMENT
'数学成绩',
  `seat` tinyint(2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
insert into `stuinfo`
values(null,'Tom',1,22,'东京',1),
(null,'Jack',1,23,'西京',2),
(null,'maria',2,20,'南京',3),
(null,'Michelle',2,20,'北京',4);

insert into `score` values(1,80,90,1),
(2,81,91,2),(3,82,92,3),(4,83,93,4);
```

1.存储过程(procedure)

语法:

```
create procedure 存储过程名(参数,…)
begin
    //代码
end//
```

注意：存储过程中有很多的SQL语句，SQL语句的后面为了保证语法结构必须要有分号（;），但是默认情况下分号表示客户端代码发送到服务器执行。必须更改结束符

通过delimiter指令来跟结束符

```
delimiter // #将结束字符定义为//(原来是;)
```

(1)创建存储过程

```
#简单的
create procedure pro_1()
select * from stuinfo;
//


#如果存储过程中就一条SQL语句，begin…end两个关键字可以省略。

#调用存储过程
call pro_1()//
```

```
#包涵多条sql语句的
#in代表输入参数,可以省略
#return
#procedure方便大型语句的查询;在创建成功以后,直接进行了语法的检查;
create procedure pro_2(in param int)
begin
    select * from stuinfo where sid=param;
    select * from score where sid=param;
end//

#调用
call pro_2(5)//
```

(2)参数的类别

在存储过程中，没有return，如果需要返回值，通过输出参数来实现
在MySQL中，参数分为3类，输入参数（in），输出参数（out）,输入输出参数（inout），默认情况下是是输入参数（in）

(3)删除存储过程

语法:drop procedure [if exists] 存储过程名

```
drop procedure if exists pro_1//
```

(4)查看存储过程的信息

```
show create procedure pro_2\G
#查看存储过程
show procedure status where db ='demo'\G
```

(5)局部变量

语法：declare 变量名 数据类型 default [初始值]
通过:select ...into…或set命令给变量赋值

例题通过sid查询姓名和年龄

```
create procedure pro_3(id int)
begin
     declare name varchar(10);
     declare sexx char(10);
```

```
    select sname,sex into name,sexx from
stuinfo where sid=id;
    select name,sexx from dual;
end //
#调用pro_3
call pro_3(3)//

#注意:声明的变量名不能和列名(字段名)同名

create table_name like table;

insert into stuinfo1 select * from
stuinfo//
```

例题:查找同桌

```
create procedure pro_4(name varchar(32))
begin
    declare stuseat tinyint;
    select seat into stuseat from stuinfo
where sname=name;
    select * from stuinfo where
seat=stuseat+1 or seat=stuseat-1;
end //

#调用

call pro_4('小芳')//
```

```
#统计大于21岁的人员占有多大的比例
create procedure pro_demo(sage tinyint)
begin
    declare cot int;
    select count(sid) into cot from stuinfo
where age>sage;
    select concat(cot/count(sid)*100,'%')
from stuinfo;
end//

call pro_demo(21)//
```

```
#通过set给变量赋值
create procedure pro_5(in num1 year,in num2
year,in name varchar(32))
begin
    declare num int default 0;
    set num=num2-num1; #得到年龄
    update stuinfo set age=num where
sname=name;
    select * from stuinfo where sname =
name;
end//

call pro_5(1996,2018,'小芳')//
```

(6)全局变量(用户变量)

全局变量前面必须有一个@，全局变量的数据类型取决于变量的值。如果一个全局变量没有赋值，他的数据类型为null。

```
set @name='小芳'//
select * from stuinfo where sname=@name//
```

(7)系统变量

通过两个@开头的都是系统变量

```
 select @@version from dual//
```

系统命令	作用
@@version	版本号
current_date	当前日期
current_time	当前时间
current_timestamp	当前日期和时间

(8)带有输出参数的存储过程

```
#带有out关键字的参数,在存储过程运行结束以后,默认返回
create procedure pro_6(in num int,out result int)
begin
    set result=num*num;
end//

#调用
#@result 接受返回值
call pro_6(6,@result)//
select @result from dual//
```

```
create procedure pro_demo1(sage tinyint,out res int)
begin
    select count(sid) into res from stuinfo where age>sage;
end//

call pro_demo1(21)//
```

(9)带有输入输出参数的存储过程

```
create procedure pro_7(inout num int)
begin
     set num=num*num;
end //

#调用

set @num=10//
call pro_7(@num)//
select @num from dual//
```

```
create procedure pro_demo2(inout res int)
begin
    select count(sid) into res from stuinfo
where age>res;
end//


set @res = 21
call pro_demo2(@res)//
```

2.SQL编程(了解)

(1) if-elseif-else语句

```
#语法:
if 条件 then
    //代码1
elseif 条件 then
    //代码2
else
    //代码3
end if;
```

```
create procedure pro_8(in grade int)
begin
     if grade=1 then
        select '金牌会员' as '等级';
     elseif grade=2 then
        select '普通会员' as '等级';
     else
         select '游客' as '等级';
     end if;
end //
#调用
call pro_8(3)//
```

(2) case-when语句

```
create procedure pro_9(in num int)
begin
    #需要做判断的变量
     case num
          when 1 then select '杀马特' as '气质';
          when 2 then select '屌丝' as '气质';
          when 3 then select '正常人' as '气质';
          when 4 then select '贵族' as '气质';
          else select '输入不正确' as '气质';
     end case;
end //

call pro_9(0)//
```

```
#显示学员的学号、姓名、性别、语文成绩、等级

select sid,sname,age,case
       when age<18 then '未成年'
       when age>=18 and age<60 then '成年'
       else '人精'
end as 'sign' from stuinfo//
```

(3)loop循环

```
#loop遇到leave退出
create procedure proc(in num int)
```

```
begin
     declare total int default 0;
     declare i int default 0;
     sign:loop
         set total=total+i;
         set i=i+1;
         if i>=num then
            leave sign;# leave=break
         end if;
     end loop;
     select total from dual;
end //

call proc(100)//
#如果没有设置标签名,leave loop
#sign是循环名,用于结束循环,可以自己随意取名字
```

(4)while循环

```
#语法:
while 条件 do
    //代码
end while
```

```
create procedure pro_11(in num int)
begin
     declare total int default 0;
     declare i int default 0;
     while num>=i do
           set total=total+i;
           set i=i+1;
     end while;
     select total from dual;
end //

call pro_11(100)//
```

(5)repeat循环

```
#语法
repeat
    代码
    until 条件     -- 直重复到条件为true才结束
end repeat
```

```
create procedure pro_12(in num int)
begin
     declare total int default 0;
     declare i int default 0;
     repeat
           set total=total+i;
           set i=i+1;
           until i>num
     end repeat;
     select total from dual;
end //

call pro_12(100)//
```

(6)leave和iterate

leave类似于break，iterate类似于continue

```
create procedure pro_13()
begin
     declare i int default 0;
     sign:while i<5 do
           set i=i+1;
           if(i=3) then
                 #leave sign;    -- 类似于
break
                 iterate sign;     -- 类似
于continue
           end if;
           select i from dual;
```

```
    end while;
end //

call pro_13()//
```

3.MySql函数

内置函数

(1).数字类

语句	含义
select rand() from dual;	随机数
select * from stuinfo order by rand();	随机排序
select round(5.6);	四舍五入
select ceil(5.3);	向上取整
select floor(5.6);	向下取整

(2).大小写转换

语句	含义
select ucase('i am lyb');	大写
select lcase('I AM LYB');	小写

(3).截取字符串

语句	含义
select left('abcdefg',3);	截取左边的3位
select right('abcdefg',3);	截取右边3位
select substring('abcdefg',2,3);	从第2位开始取3个字符，起始位置从1开始

(4).字符串拼接

```
select concat(sid,sname,age,sex,city) from stuinfo;
```

(5).coalesce(str1,str2):如果str1不为null则显示str1，否则显示str2

```
#指定的条件为Null,我们会自动补齐
select sid,sname,coalesce(ch,0) ch,coalesce(math,0) math from stuinfo left join score using(sid);
```

(6).length(字节长度)、char_length(字符长度)、trim(去两边空格)、replace(替换)

```
select length('千锋');

select char_length('千锋');

select length(trim(' 千锋 '));

select replace('pgone','one','two');
```

(7).时间戳

```
select unix_timestamp();
```

(8).将时间戳转成当前时间

```
select from_unixtime(unix_timestamp());
```

(9).获取当前时间

```
select
now(),year(now()),month(now()),day(now()),h
our(now()), minute(now()),second(now())\G

#现在时间,年,月,日,时,分,秒
```

(10).dayname(),monthname(),dayofyear()

```
select dayname(now()) as `星期
`,monthname(now()) as `月份
`,dayofyear(now()) as `本年第几天`;
```

(11).datediff(结束日期，开始日期)

```
#例题计算自己活了多少天
select datediff(now(),'2000-10-01');
```

(12).加密

```
select md5('@123456.');
select password('123456')
```

3.自定义函数

```
#语法:
Create function 函数名(形参) returns 返回的数
据类型
begin
    //函数体
end
```

```
#第一步
delimiter //

#不带参数的函数
create function myfun() returns varchar(32)
begin
```

```
    return 123;
end//

#调用函数
select myfun()//

#Linux中的mysql不支持函数
#先查看是否支持
show variables like
'log_bin_trust_function_creators';
#进入/etc/my.cnf
#放在[mysqld]
log_bin_trust_function_creators=1
#写好以后重启mysql服务器
service mysqld restart
```

```
#带参数
create function myfun_1(num1 int,num2 int)
returns int
begin
    declare num int default 0;
    set num=num1+num2;
    return num;
end //

select myfun_1(100,200)//


#删除函数
drop function myfun_1//
```

```
#自己封装 首字母大写
create function firstup(str varchar(1024))
returns varchar(1024)
begin
    return
concat(ucase(substr(str,1,1)),substr(str,2)
);
end//
```

4.触发器

```
1、触发器是一个特殊的存储过程
2、不需要直接调用，在MySQL自动调用的
3、是一个事务，可以回滚
```

(1)触发器的类型(触发事件)

```
1、insert触发器
2、update触发器
3、delete触发器
```

(2)创建触发器

```
#语法:
create trigger 触发器名 触发时间[before|after]
触发事件 on 表名 for each row
begin
    //代码
end//
```

(3)new表和old表

1、这两个表是个临时表
2、当触发器触发的时候在内存中自己创建，触发器执行完毕后自动销毁
3、它们的表结构和触发器触发的表的结构一样
4、只读，不能修改

stuinfo curd

打开文件,内存中需要加载,会随即分配一个空间用来保存文件的所有数据,->old 6

在新的一轮操作后,内存会生成新的空间,这个空间里面保存了新的数据变化->new 7

(5)insert触发器

```
#在stuinfo中插入一个值,就会自动在stumarks中插入一条数据
#after insert 表示的是在insert动作执行完毕以后触发
#on stuinfo for each row  针对的stuinfo表,并且可以读取到每一行的变化
#触发器中定义的局部变量不能与表中的字段名一致,否则会发生字段识别问题(识别不出到底是字段,还是变量)
create trigger trig1
after insert on stuinfo
for each row
begin
```

```
    #获取id
     declare sidno int default 0;
     #定义的语文成绩
     declare nch int default 0;
     #定义的数学成绩
     declare nma int default 0;
     #座位号
     declare nseat int default 0;
     #sudno =  new表中sid(刚刚插入的新数据的id)
     set sidno=new.sid;
     #获取临时new表当中的座位号
     set nseat=new.seat;
     insert into score set
sid=sidno,ch=nch,math=nma,seat=nseat;
end //

insert into stuinfo values(null,'鸡
哥',1,23,'安庆',16)//
```

(6)update触发器

```
create trigger trig2
after update on stuinfo for each row
begin
    declare sidno int default 0;
    declare seatno int default 0;
    set seatno=new.seat;
    set sidno =new.sid;
    update score set seat=seatno where sid
= sidno;
end //

select ((select max(seat) from
stuinfo)+1)//
update stuinfo set seat=17 where sid=8//
```

(7)delete触发器

```
create trigger trig3
after delete on stuinfo for each row
begin
     declare sidno int default 0;
     set sidno =old.sid; #删除了新表里面就没有
了,只能从老表里面拿
     delete from score where sid=sidno;
end //

delete from stuinfo where sid =8//

#触发器能做钩子函数
```

(8)查看 和 删除 触发器

```
show triggers\G

drop trigger if exists trig1//
```

5.用户管理

```
[mysqld]
skip--grant--tables
#skip--grant--tables 跳过登陆验证(MYSQL服务器开起中)
```

(1)创建用户

```
语法：create user ‘用户名’@’允许登录的主机地址’ identified by 密码
```

```
#%代表数据库的库名
create user 'hal'@'%' identified by '123456';
```

(2)删除用户

```
语法：drop user 用户
```

```
drop user ruidong;
```

(3)增加用户权限

```
#将python的所有表的select权限付给用户
#grant select on 运行使用的数据库.允许使用的表
to 'hal'@'%';
grant select on database_name.table_name to
'hal'@'%';

#将所有数据库中所有表的所有权限付给用户
grant all privileges on *.* to 'hal'@'%';

#创建用户并授权
grant all privileges on *.* to 'hal'@'%'
identified by '123456' with grant option;

#创建好用户以后,刷新mysql用户权限表
flush privileges ;(linux ,mac)
```

```
revoke select on python.* from
'ruidong'@'%';   #删除select权限
revoke all privileges on *.* from
'ruidong'@'%'; #删除所有权限
```

(4)mysql57忘记密码

```
1、首先停止mysql服务进程：
service mysqld stop
2.#然后编辑mysql的配置文件my.cnf(如果是windows的话找到my.ini)
vim /etc/my.cnf
3.#找到 [mysqld]这个模块：
#在最后面添加一段代码
skip-grant-tables    ##忽略mysql权限问题，直接登录
#然后保存 :wq!退出
#启动mysql服务：
service mysqld start
```

```
#直接进入mysql数据库：
mysql
#选择mysql数据库
use mysql;
#对user表的root用户进行密码修改
update mysql.user set
authentication_string=password('123456')
where user='root' and Host = 'localhost';

#特别提醒注意的一点是，新版的mysql数据库下的user表中已经没有Password字段了
#而是将加密后的用户密码存储于
authentication_string字段


#执行刷新
 flush privileges;
```

```
#exit退出mysql
exit;
#启动服务
service mysqld start
```