Day 05 MySQL存储过程

一、MySQL存储过程

1.存储过程概述

1) 存储过程介绍

•存储过程相当于是MySQL语句组成的脚本

- 指的是数据库中保存一系列SQL命令的集合

- 可以在存储过程中使用变量、条件判断、流程控制等

2) 存储过程优点

•提高性能

•减轻网络负担

•防止对表的直接访问

•避免重复写SQL操作

2.基本使用

1) 创建存储过程

> delimiter //

> create procedure say()

-> begin

-> select \* from studydb.user where name="root";

-> end

-> // 结束存储过程

> delimiter ;

delimiter关键字声明当前段分隔符

MySQL默认以";"为分隔符，没有声明分隔符，编译器会把存储过程当成SQL语句进行处理，则存储过程的编译过程会报错。

EXAMPLE:

> delimiter //

> create procedure say1()

-> begin

-> select \* from db9.user;

-> select \* from mysql.user where user="root";

-> end

-> //

> delimiter ;

2) 查看存储过程

•查看存储过程

•) 方法1

> show procedure status;

•) 方法2

> select db,name,type from mysql.proc where name="存储过程名";

> desc mysql.proc;

> select body from mysql.proc where name="say1"\G //查看存储过程代码

3) 调用/删除存储过程

•调用存储过程

- call 存储过程名();

> call say1();

> call say;

•删除存储过程

- drop procedure 存储过程名

3.存储过程进阶

·编写功能体代码时，可以使用变量、条件判断、流程控制（if,循环）、算数计算、sql命令

1) 变量类型

·变量的种类：全局变量|会话变量|用户变量|局部变量

·) 会话变量&全局变量：会话变量和全局变量叫做系统变量，使用set命令定义;全局变量的修改会影响到整个服务器，但是对会话变量的修改，只会影响到当前的会话。

·) 用户变量：在客户端连接到数据库服务的整个过程中都是有效的，当当前连接断开后所有用户变量失效。

定义: set @变量名=值;

输出: select @变量名;

·) 局部变量：存储过程中的begin/end。其有效范围仅限于该语句块中，语句块执行完毕后，变量失效。

declare专门用来定义局部变量。

·) 注意：局部变量和参数变量调用时，变量名前不需要加@

EXAMPLE:

> delimiter //

> create procedure p2()

begin

declare x int default 77;

declare y char(10);

set y="yaya";

select x;

select y;

end

//

> delimiter ;

> call p2();

> select @x;

> select @y;

> show global variables; //查看全局变量

> show session variables; //查看会话变量

> set session sort\_buffer\_size = 40000; //设置会话变量

> show session variables like "sort\_buffer\_size"; //查看会话变量

> show global variables like "%关键字%"; //查看全局变量

> set @y = 3; //用户自定义变量，直接赋值

> select max(uid) into @y from db9.user; //使用sql命令查询结果赋值

> select @y;

> select @@hostname;

> delimiter //

> create procedure p3()

begin

declare x int default 77;

select x;

select max(uid) into x from db9.user;

select x;

end

//

> delimiter ;

> call p3;

2) 参数类型

·调用参数时，名称前也不需要加@

> create procedure 名称(

类型 参数名 数据类型,类型 参数名 数据类型

)

in 输入类型参数 作用是给存储过程传值必须在调用存储过程时赋值，在存储过程中该参数的值不允许修改;默认类型是in

out 输出类型参数 该参数可在存储过程内部被改变，并可返回

inout 输入输出型参数 调用时指定，并且可被改变和返回

指定存储过程参数的格式：参数类型 参数名 类型(宽度)

EXAMPLE:

> delimiter //

> create procedure say(in username char(10)) //定义in类型的参数变量username

begin

select username;

select \* from user where name=username;

end

//

> delimiter ;

delimiter //

> create procedure p5(in username char(20))

begin

select name from db9.user where name=username;

end

//

delimiter ;

> call p5(); //没有传值，报错

> call p5("tom");

> call p5("bob");

> set @name="tom";

> call p5(@name);

> delimiter //

> create procedure p6( out num int(2) )

begin

select num;

set num=7;

select num;

select count(name) into num from db9.user where shell!="/bin/bash";

select num;

end

//

> delimiter ;

> call p6();

> call p6(88); //报错，out和inout类型参数不能直接输入，要通过传递参数的形式传值

> set @x=88;

> select @x;

> call p6(@x);

> set @x=1;

> select @x;

> call p6(@x);

执行结果：

null

7

40

> delimiter //

> create procedure p7( inout num int(2) )

begin

select num;

set num=7;

select num;

select count(name) into num from from db9.user where shell!="/bin/bash";

select num;

end

//

> delimiter ;

> set @x=9;

> select @x;

> call p7(@x);

执行结果：

9

7

40

3) 算数运算

·算数运算符号

+ 加法运算

- 减法运算

\* 乘法运算

/ 除法运算

DIV 整除运算

% 取模/取余

EXAMPLE:

> deliomiter //

> set @z=1+1; select @z;

> create procedure p4()

begin

declare x int;

declare y int;

declare z int;

select count(shell) into x from db9.user where shell="/bin/bash"

select count(shell) into y from db9.user where shell="/sbin/nologin";

set z=x+y;

select z;

end

//

> delimiter ;

4.流程控制

1) 条件判断

· 与查询命令select的条件判断语法相同，此处的条件判断是给流程控制结构做执行条件

= 等于

>,>= 大于，大于等于

<,<= 小于，小于等于

!= 不等于

Between ... and ... 在...和...之间

· 逻辑比较、范围、空、非空、模糊、正则

OR、AND、! 逻辑或与非

IN...、NOT IN ... 在..范围内、不在...范围内

IS NULL 字段的值为空

IS NOT NULL 字段的值不为空

LIKE 模糊匹配

REGEXP 正则匹配

EXAMPLE:

> select name,gid,uid from db9.user where uid=gid;

> select name,gid,uid from db9.user where uid=3;

> select name,gid,uid from db9.user where uid is null;

2) 选择结构/顺序结构

·当"条件成立"时执行命令序列

·否则，不执行任何操作

if 条件测试 then

代码...

...

end if;

·当"条件成立"时执行代码1

·否则，执行代码2

if 条件测试 then

代码1...

...

else

代码2...

...

end if;

EXAMPLE：

> drop procedure if exists say;

> delimiter //

> create procedure say(in x int(1))

begin

if x <= 10 then

select \* from userdb.user where id <= x;

end if;

end

//

> delimiter ;

> call say(1); #条件判断成立

> delimiter //

> create procedure p8( in num int(2) )

begin

if num <= 10 then

select \* from db9.user where id <= num;

end if;

end

//

> delimiter ;

> call p8(3);

> call p8(7);

> call p8(11);

> select \* from db9.user where id<=3;

> set @x=7

> call p8(@x);

> delimiter //

> create procedure p9( in num int(2) )

begin

if num is null then

select \* from db9.user where id=2;

else

select \* from db9.user where id <= num;

end if;

end

//

> delimiter ;

3) 循环结构

·条件式循环

- 反复测试条件

- 只要成立就执行命令序列

while 条件判断 do

循环体

...

end while;

·条件式循环

- 无循环条件

loop

循环体

...

end loop;

·条件式循环

- until条件判断，成立时结束循环

- 不管符不符合条件都先执行一次

repeat

循环体

until 条件判断

end repeat;

·控制语句

循环结构控制语句，控制循环结构的执行。

- LEAVE 标签名 //跳出循环，类似shell中的break

- ITERATE 标签名 //放弃本次循环，执行下一次循环，类似shell中的continue

EXAMPLE:

> delimiter //

> create procedure say()

begin

declare i int;

set i=1;

while i <= 5 do

set i=i+1;

end while;

select i; //输出最终结果

end

//

> delimiter ;

> delimiter //

> create procedure p10()

begin

declare i int(2);

select count(uid) into i from db9.user;

set j=1;

while j <= i do

if j%2=0 then

select \* form db9.user where id=j;

end if;

set j=j+1;

end while;

end

//

> delimiter ;

> delimiter //

> create procedure say2()

begin

declare i int;

set i=1;

loop

select i;

set i=i+1;

end loop;

end

//

> delimiter ;

> call say2(); #不按ctrl+c结束 会一直输出变量i的值

> delimiter //

> create procedure p11()

begin

declare j int(2);

set j=1;

loop

select j;

set j=j+1;

end loop;

end

//

> delimiter ;

> delimiter //

> create procedure p12()

begin

declare j int(2);

set j=1;

loop

select j;

set j=j+1;

until i=6

end repeat;

end

//

> delimiter ;

> call p12;

> delimiter //

> create procedure p13()

begin

declare i int(2);

set i = 1;

loab1:loop

select i;

set i = i+1;

leave loab1;

end loop;

end

//

> delimiter ;

> delimiter //

> create procedure p14()

begin

declare i int(2);

set i = 1;

loab1:loop

select i;

set i = i+1;

iterate loab1;

end loop;

end

//

> delimiter ;

> drop procedure p15;

> delimiter //

> create procedure p15()

begin

declare i int(2);

set i=0;

loab1:while i <= 10 do

set i = i+1;

if i=7 then

ITERATE loab1;

end if;

select i;

end while;

end

//

> delimiter ;

> call p15;