

day11 MySQL

一.内容简介

1. 存储过程(函数);
2. 触发器;
3. MySQL 权限问题;
4. MySQL 的备份与恢复.

二.存储过程(函数)

1.概念:

带有逻辑判断的 sql 语句,就是所谓的存储过程(函数).
也就是说可以给 sql 语句添加 if....else...while 等语句.

2.特点:

- ①.执行效率高:存储过程(函数)是在 MySQL 的服务器端来执行的语句,效率就比在 Java 中结合 SQL 判断效率高.
- ②.可移植性差:不同的数据库软件,支持的存储过程(函数)的语法不同,所以不同的数据库软件之间,存储过程不能直接移植.

3.存储过程语法

delimiter \$

create procedure 存储过程名(参数)

begin

.....具体的代码

end \$

delimiter:声明,\$是存储过程开始与结束的标记.

4.存储函数语法

delimiter \$

create function 存储函数名(参数) return 返回值类型

begin

...具体的代码

end \$

5. 具体案例

①. 定义不带参数的存储过程

```
-- 这是客户端的注释  
  
-- delimiter $声明存储过程的开始与结束标记  
DELIMITER $  
  
-- 创建存储过程, 指定存储过程的名称, 注意: 名称后面要带括号()  
CREATE PROCEDURE first_pro()  
  
-- 表示开始执行存储过程  
BEGIN  
  
    -- 随便定义sql语句  
    SELECT * FROM emp;  
  
-- 表示存储过程结束  
END $
```

调用存储过程:

```
-- 调用存储过程  
CALL first_pro();
```

执行结果:

	empno	ename	job	mgr	hiredate	sal	comm	deptno
<input type="checkbox"/>	7369	SMITH	CLERK	7902	1980-12-17	800.00	(NULL)	20
<input type="checkbox"/>	7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
<input type="checkbox"/>	7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
<input type="checkbox"/>	7566	JONES	MANAGER	7839	1981-04-02	2975.00	(NULL)	20

②. 存储过程的参数类型

输入参数: IN

IN id INT:

IN: 表示要输入一个参数;

id: 参数名称;

INT:参数类型.

输出参数:OUT

OUT msg char(20)

输入输出 :INOUT

③.定义一个带有输入参数的存储过程

-- 定义带参数的存储过程, 根据部门编号查询部门信息
DELIMITER \$

CREATE PROCEDURE second_pro(IN dNo INT)

BEGIN

-- 注意:存储过程中的语法, 大小写不区分. deptno等同于deptNO
SELECT * FROM dept WHERE deptno = dNo;

END \$

调用存储过程:

CALL second_pro(30);

结果:

	deptno	dname	loc
<input type="checkbox"/>	30	SALES	CHICAGO

④.删除存储过程

-- 删除存储过程

DROP PROCEDURE second_pro;

⑤.带有输出参数的存储过程

```
DELIMITER $  
  
-- 定义输出参数  
CREATE PROCEDURE third_pro(OUT msg CHAR(20))  
  
BEGIN  
  
    SET msg='被输出的结果';  
  
END $
```

执行结果:

```
-- @str 用来接收输出结果  
CALL third_pro(@str);  
  
-- select @str显示结果  
SELECT @str;
```

@str
被输出的结果

⑥.带有输入和输出参数的例子

```
-- 根据id, 查询某个学生的姓名  
DELIMITER $  
  
CREATE PROCEDURE fourth_pro(IN sid INT ,OUT sname CHAR(20))  
  
BEGIN  
  
    SELECT NAME INTO sname FROM student WHERE id=sid;  
  
END $  
  
CALL fourth_pro(1,@NAME);  
  
SELECT @NAME;
```

@NAME
Tom

⑦.输入输出参数 INOUT

```
-- 输入输出参数
DELIMITER $

CREATE PROCEDURE five_pro(INOUT n INT)
BEGIN
    -- 查看变量值
    SELECT n;

    SET n=1000;
END $

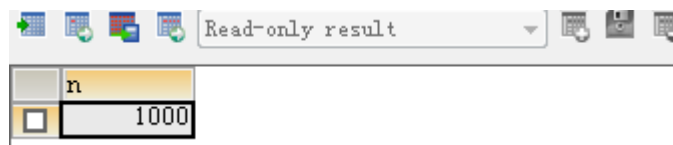
-- 给输入参数赋值
SET @n=200;
|
CALL five_pro(@n);

-- 利用输出参数取值
SELECT @n;
```

结果一:

n=200

结果二:



Read-only result	
n	1000

⑧.存储过程中使用 if...then...elseif...end if;语句.

```
DELIMITER $

CREATE PROCEDURE six_pro(IN val INT ,OUT msg CHAR(30))
BEGIN
    IF val=1 THEN
        SET msg='星期一';
    ELSEIF val=2 THEN
        SET msg='星期二';
    ELSEIF val=3 THEN
        SET msg='星期三';
    END IF;
    -- 注意:if...then...elseif...endif...的写法,最后要有分号!
END $

CALL six_pro(2,@msg);|

SELECT @msg;
```

⑨.存储过程中使用循环 while

```
DELIMITER $

CREATE PROCEDURE seven_pro(IN n INT ,OUT result INT)
BEGIN
    -- declare:声明变量
    -- default:指定默认值
    DECLARE i INT DEFAULT 1;
    DECLARE vSum INT DEFAULT 0;

    WHILE i<=n DO

        SET vSum=vSum+i;

        SET i=i+1;

    END WHILE;

    SET result=vSum;

END $
```

```
CALL seven_pro(100,@result);  
SELECT @result;
```

三.触发器

1.概念:

在操作一张表的时候,与此同时要触发另外的一些行为和动作,这样的操作就可以使用触发器.比如当操作一个表的时候,触发另一表来进行记录操作.

比如有个员工表,当往员工表中插入/update/delete 等操作的时候,与此同时往另外一个 log 表插入一条记录.

2.语法:

```
-- TRIGGER:触发器  
-- AFTER:指明触发器的执行时机.  
-- insert on 表名:当往表中添加数据的时候....  
-- FOR EACH ROW :在行级别上触发触发器.  
-- insert....是触发器被触发后要执行的动作.
```

```
CREATE TRIGGER stu_log_trg AFTER INSERT ON student FOR EACH ROW  
INSERT INTO myLog (msg) VALUES('student表中添加了一条记录');
```

```
CREATE TRIGGER stu_log_update AFTER UPDATE ON student FOR EACH ROW  
INSERT INTO myLog (msg) VALUES('修改了student表');
```

```
CREATE TRIGGER stu_log_delete AFTER DELETE ON student FOR EACH ROW  
INSERT INTO myLog (msg) VALUES('删除了student表的数据');
```

结果:

	id	msg
<input checked="" type="checkbox"/>	1	student表中添加了一条记录
<input checked="" type="checkbox"/>	2	修改了student表
<input checked="" type="checkbox"/>	3	删除了student表的数据

四.视图 View

1.概念:

视图也是一个“表”,只不过是一个“虚拟的表”.

视图是基于 SQL(select)语句的结果集的可视化的表.

2.作用:

提高查询的效率,一般只用于查询.

3.语法

create View view_name as

select 列名 from 表名

where....

注意:

as 后面的查询语句,可以是我们之前的任意一种 select 语句.

单表,多表查询,多表连接等都可以.

案例:创建一个视图

```
CREATE VIEW emp_view AS  
SELECT ename FROM emp;
```

查询视图:

```
SELECT * FROM emp_view;
```

	ename
<input type="checkbox"/>	SMITH
<input type="checkbox"/>	ALLEN
<input type="checkbox"/>	WARD
<input type="checkbox"/>	JONES
<input type="checkbox"/>	MARTIN
<input type="checkbox"/>	BLAKE

五.MySQL 权限问题

1.mysql 数据库

存放 MySQL 自身的配置信息.

User 表:用来存储 MySQL 的用户信息的表.

2.password()函数

将密码进行 md5 加密.

案例:

```
SELECT PASSWORD('root');|
```

	password('root')
<input type="checkbox"/>	*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B

3.往 user 表中添加一个用户.

```
INSERT INTO USER (HOST,USER,PASSWORD) VALUES('localhost','tom',PASSWORD('123'));
-- 当往user表中插入了一个新用户之后,得做一个额外的操作,user表才会进行真正的更新!
```

注意:

当对 user 表进行了 insert/update/delete 等操作后,必须刷新权限,否则会导致以下错误!

```
FLUSH PRIVILEGES;
```

```
GRANT SELECT ON db01.emp TO 'tom'@'localhost' IDENTIFIED BY '123';
```

错误信息:

错误码: 1133

Can't find any matching row in the user table

3.grant 命令

案例:

给 localhost 服务器中的 tom 分配查询权利,而且 tom 只能查询 db01 数据库中的 emp 表.

```
GRANT SELECT ON db01.emp TO 'tom'@'localhost' IDENTIFIED BY '123';
```

错误信息:

错误码: 1142
DELETE command denied to user 'tom'@'localhost' for table 'emp'

六.MySQL 数据库的备份和恢复

1.数据库备份

```
C:\Windows\system32>mysqldump -uroot -p db01 > C:/db01.sql  
Enter password: ***
```

注意:

- ①.以管理员身份打开命名行;
- ②.整个的备份语句后面不要带分号;

2.数据库恢复

```
C:\Windows\system32>mysql -u root -p db02 < C:/db02.sql
Enter password: ***
```

注意:

- ①.恢复数据库用的命名是 **mysql**,不是 **mysqldump**!
- ②.恢复数据库之前,要手动的在数据库的客户端中建立一个同名的空白数据库.

七.数据库设计规范

数据库设计的 3 大范式

1. 第一范式:

表中的每个字段都必须是不可再分割的独立单元.

student----->name 不符合第一范式!

student-->old_name 符合第一范式!

2. 第二范式:

在第一范式的基础之上,要求每张表都只能表达一个意思.

表的每个字段都应该与表的主键有依赖关系.

员工表(emp1): 员工编号 员工姓名 部门名称 订单编号
不符合第二范式

员工表(emp2): 员工编号 员工姓名 部门名称 符合第二

范式

3. 第三范式:

在第二范式的基础之上,其他的字段和主键直接只能有直接依赖关系.

员工表(emp1): 员工编号 员工姓名 部门名称

部门表:部门编号 部门名称

第一个案例:不符合第三范式.因为用部门名称会产生数据的冗余.

员工表(emp2): 员工编号 员工姓名 部门编号

部门表:部门编号 部门名称

第二个案例:符合第三范式.