2020/08/25_数据库_第4课_存储过程_事务、MySQL API的使用

笔记本： 数据库
创建时间： 2020/8/25 星期二 10:56
作者： ileemi

存储过程

将sql语法提前编译好，类似于C语言函数。将查询语法提前封装好。
什么是存储过程：
存储过程是一组预先编译好的Transact-SQL代码。它可以对数据库进行查询和更新。
存储过程类似计算机高级语言的过程，也可以带参数并返回处理结果。

使用存储过程的优点：

- 执行速度快
- 有利于应用程序执行sql语句
- 减少网络通信量

存储过程的分类

- 系统存储过程
- 用户自定义存储过程
- 扩展存储过程

系统存储过程：
select DATEBASE() -- 获取当前数据库名称
select USER() -- 当前操作用户

> 创建存储过程：

```
CREATE PROCEDURE FUN1
BEGIN
语法
END
```

注意语法后的 分号

```
67
68  delimiter //
69  CREATE PROCEDURE FUN1()
70  BEGIN
71    SELECT *FROM T_STUDENT;
72  END//
73  delimiter ;
74
75
```

过程函数创建完成后，其会保存到数据库中（函数）。

调用存储过程：
CALL FUN1();

存储可以传参，存储过程名后加参数：
IN -- 传入参数
OUT --- 返回值，传出参数

```
78  delimiter //
79  CREATE PROCEDURE FUN2(IN ID VARCHAR(255), IN NAME VARCHAR(255))
80  BEGIN
81    SELECT *FROM T_STUDENT WHERE STU_ID = ID AND STU_NAME = NAME;
82  END//
83  delimiter ;
84
```

```
88  BEGIN
89    SELECT COUNT(*) INTO count  FROM T_STUDENT;
90  END//
91  delimiter ;
92
93  -- 调用存储过程
94  CALL FUN1();
95  CALL FUN2('S001', '张三');
96  CALL FUN3(@count);SELECT @count;
97
```

> SHOW 命令

SHOW CREATE PROCEDURE xxx -- 显示存储过程
SHOW CREATE DATEBACE xxx -- 显示创建的数据库具体信息

事务 TRANSACTION

增加班级 增加学生 （存在增加班级失败）

insert into T_CLASS VALUES('C005', '科锐37班');
insert into T_STUDENT VALUES('S100', 'xxx', 'C005');
insert into T_STUDENT VALUES('S101', 'xxx', 'C005'); -- 数据插入失败
insert into T_STUDENT VALUES('S102', 'xxx', 'C005');

select * from T_STUDENT;
select * from T_CLASS;

数据插入失败，应该将成功插入的数据撤销，回滚（删除 -- 增加，增加 -- 删除）。
数据库具有**事务日志**，根据请求进行数据回滚。

> 事务的操作：
> 开始事务（START TRANSACTION）：添加数据前
> 回滚事务（ROLLBACK）： 添加的数据进行删除
> 提交事务（COMMIT） ： 提交后，操作不能回滚

```
-- try {
  START TRANSACTION;  -- 开始事务
  insert into T_STUDENT VALUES('S100', '阿斯蒂芬', 'C005');
  insert into T_STUDENT VALUES('S001', '按位', 'C005');
  insert into T_STUDENT VALUES('S101', '爱上', 'C005');
  COMMIT;  -- 提交
-- }
-- catch(...) {
  ROLLBACK; -- 回滚
-- }

```

使用 MySQL API

MySQL动态库的使用

> 初始化MySql库
> mysql_init()

> 连接数据库
> mysql_real_connect()

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if
(!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Failed to connect to database: Error: %s\n",
          mysql_error(&mysql));
}
```

> 设置工程字符集名称
> mysql_set_character_set

> mysql_character_set_name();

> 显示影响的行数
> mysql_affected_rows

> 操作数据库
> 增加：mysql_real_query()

> 关闭数据库
> mysql_close(&mysql)

> 查询数据

> 保存结果
> MSQL_RES* res = mysql_store_result(&mysql);

> 释放保存结果的缓冲区
> mysql_free_result(res)

> 遍历字段

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

// 得到字段的数量
num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
// 循环遍历表的字段信息
for(i = 0; i < num_fields; i++)
{
   printf("Field %u is %s\n", i, fields[i].name);
}
```

> 遍历数据

```
// 遍历数据
MYSQL_ROW row;
// 根据结果遍历行数
while ((row = mysql_fetch_row(result)))
{
  unsigned long *lengths;
  lengths = mysql_fetch_lengths(result);
  //  根据字段数量遍历行数据
  for (i = 0; i < num_fields; i++)
  {
    printf("%-15s", row[i]);
  }
  printf("\n");
}
```

MySQL 线程同步问题：

mysql_thread_init() -- 同步
mysql_thread_end() -- 终止同步
mysql_ssl_set() -- 数据加密
mysql_kill -- 强制结束当前线程（少用）