

学习目标

- * 能够PLSQL Developer远程连接Oracle数据库
 - * 绿色版
 - * 配置
 - * 能够掌握Oracle的表空间的使用
 - * 给用户创建账号--->对应着不同表空间
 - * 能够理解事务相关的概念
 - * 保证数据库数操作数据一致性
 - * 一组dml语句组成，要么一起成功，要么一起失败
 - * commit , rollback,savepoint
 - * 能够理解事务四种特性ACID
 - * A：原子性
 - * C:一致性
 - * D:持久性
 - * I：隔离性

* SQL92: Read Uncommitted (出现脏读，不可重复读，虚读),ReadCommitted(不可重复读，虚读),Repeatable(虚读),Serializable

 - * MYSQL:四种都支持
 - * Oracle:支持ReadCommitted , Serializable , Read Only
 - * MySQL：默认隔离级别：Repeatable
 - * Oracle：默认隔离级别：ReadCommitted
 - * 能够掌握使用JDBC控制数据库的事务
 - * Connection:setAutoCommitted(false);
 - * Connection:commit
 - * COnnection:setSavePonit
 - * Connection:rollback,rollback(sp)
-

* 能够PLSQL Developer远程连接Oracle数据库

* 参考 [PLSQL Developer 远程配置](#)

* 能够掌握Oracle的表空间的使用

* Oracle表空间的概述

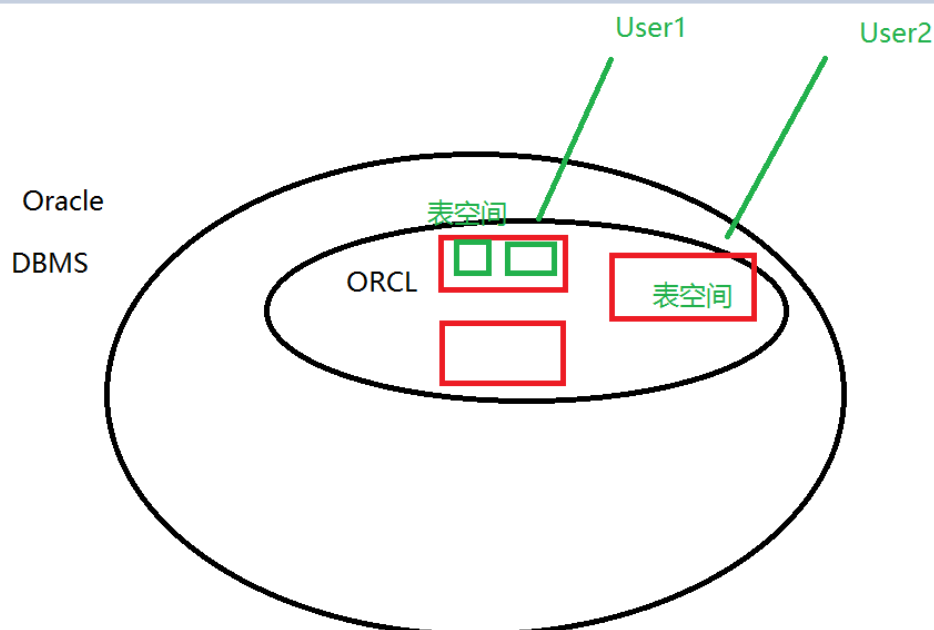
* 表空间是数据库的逻辑划分，一个表空间只能属于一个数据库。

* 所有的数据库对象都存放在指定的表空间中。但主要存放的是表，所以称作表空间。

* Oracle数据库中至少存在一个表空间，即SYSTEM的表空间。

* Oracle数据库开创性地提出了表空间的设计理念，这为Oracle数据库的高性能做出了不可磨灭的贡献。可以这么说，Oracle中很多优化都是基于表空间的设计理念而实现的。

* Oracle数据库由若干表空间构成，表空间由一到多个数据文件组成，每个数据文件只能属于同一表空间。



* Oracle表空间的分类

* 永久性表空间：一般保存表、视图、过程和索引等的数据库

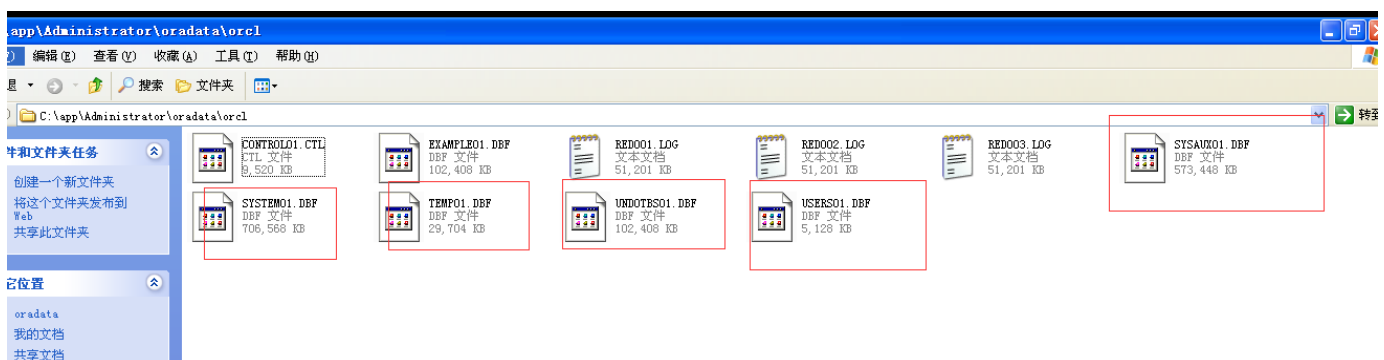
* 临时性表空间：只用于保存系统中短期活动的数据库

* 撤销表空间：用来帮助回退未提交的事务数据库

* 查看表空间

- 1 * 使用system用户登录，密码123123
- 2 * 查询表空间的记录
- 3 * SELECT * from dba_data_files;

FILE_NAME	FILE_ID	TABLESPACE_NAME	BYTES	BLOCKS	STATUS	RELATIVE_FNO	AUTOEXTENSIBLE	MAXBYTES	MAXBLOCKS	I
C:\APP\ADMINISTRATOR\ORADATA\ORCL\USERS01.DBF	4	USERS	14417920	1760	AVAILABLE	4	YES	34359721984	4194302	
C:\APP\ADMINISTRATOR\ORADATA\ORCL\UNDOTBS01.DBF	3	UNDOTBS1	104857600	12800	AVAILABLE	3	YES	34359721984	4194302	
C:\APP\ADMINISTRATOR\ORADATA\ORCL\SYSAUX01.DBF	2	SYSAUX	566231040	69120	AVAILABLE	2	YES	34359721984	4194302	
C:\APP\ADMINISTRATOR\ORADATA\ORCL\SYSTEM01.DBF	1	SYSTEM	723517440	88320	AVAILABLE	1	YES	34359721984	4194302	
C:\APP\ADMINISTRATOR\ORADATA\ORCL\EXAMPLE01.DBF	5	EXAMPLE	104857600	12800	AVAILABLE	5	YES	34359721984	4194302	



* 创建表空间

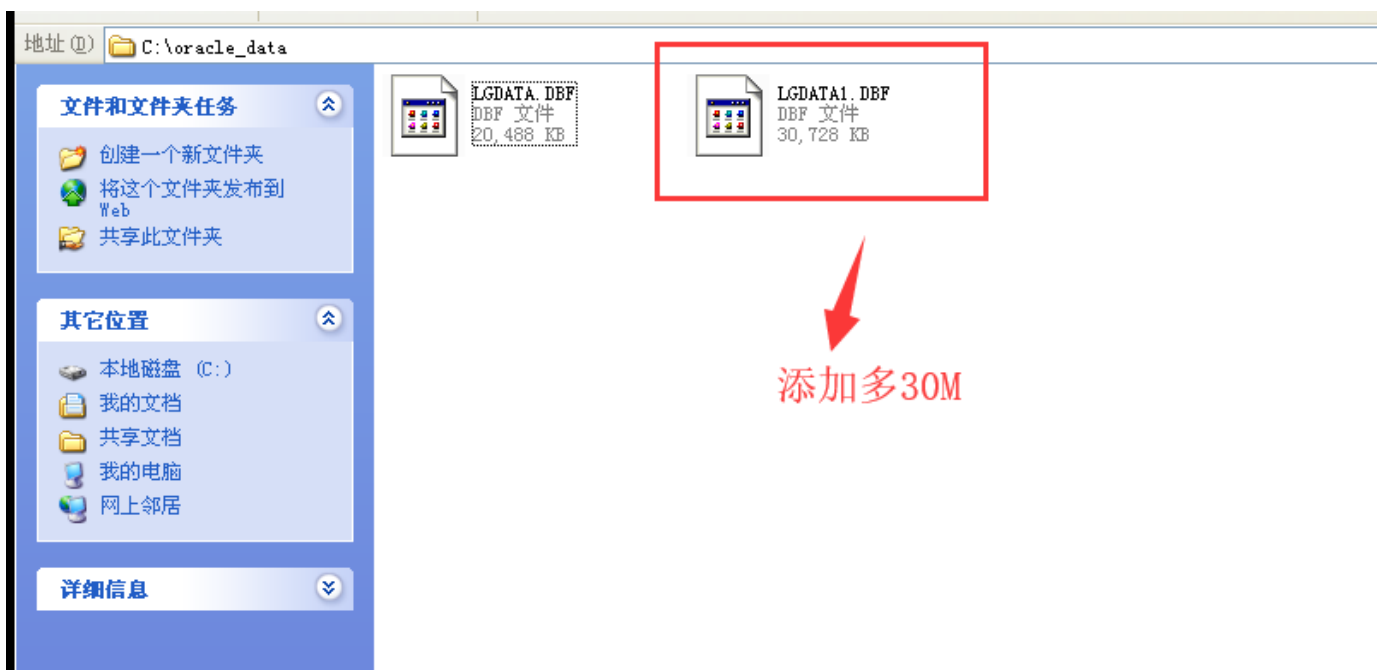
- 1 * 创建表空间
- 2 * 在C盘需要先创建oracle_data目录，以存放数据文件
- 3 * 创建表空间为lgdata，文件名字lgdata.dbf，大小为20M，可自动扩展
- 4 * create tablespace lgdata
- 5 datafile 'C:\oracle_data\lgdata.dbf' size 20M autoextend on;
- 6
- 7 * 查询表空间是否创建成功
- 8 * SELECT file_name,tablespace_name,bytes,autoextensible
- 9 FROM dba_data_files where tablespace_name='LGDATA';
- 10
- 11 * 调整表空间大小,向表空间内添加数据文件
- 12 * alter tablespace lgdata
- 13 add datafile 'C:\oracle_data\lgdata1.dbf' size 30M
- 14 autoextend on;
- 15
- 16 * 删除表空间
- 17 * drop tablespace lgdata;
- 18

```
19 * 删除表空间及数据文件
20 * drop tablespace lgdata including contents and datafiles;
21
22 * 使用表空间
23 * 创建表空间
24 * create tablespace lgdata
25     datafile 'C:\oracle_data\lgdata.dbf' size 30M
26     autoextend on next 20M;
27
28 * 创建用户使用表空间和临时
29 * create user lg identified by 123123
30     default tablespace lgdata
31     temporary tablespace temp;
32
33 * 查询用户的记录
34 * SELECT * FROM dba_users WHERE username='LG';
35 * 给用户授权
36 * GRANT connect, resource TO LG;
37 * 回收用户权限
38 * REVOKE connect, resource FROM lg; --撤销CONNECT和RESOURCE两个角色
39 * 修改用户密码
40 * alter USER lg identified by 888888;
41 * 删除用户（级联清除表空间的数据，但是文件还存在）
42 * drop user lg cascade;
43
44 * 系统用户的介绍
45 * sys用户：超级用户，主要用来维护系统信息和管理实例，以SYSDBA或SYSOPER角色登录。
46 * system用户：默认的系统管理员，拥有DBA权限，通常用来管理Oracle数据库的用户、权限和
47 * scott用户：示范用户，使用users表空间。
48
49 * 思考：
50 * 假如作为数据库管理员，需要给每个学生分配不同用户，不同空间，如何做？
51
```

* 操作部分截图



							
		FILE_NAME		TABLESPACE_NAME		BYTES	AUTOEXTENSIBLE
▶	1	C:\ORACLE_DATA\LGDATA.DBF ...		LGDATA	...	20971520	YES



* 能够理解事务相关的概念

* 事务用于保证数据的一致性，它由一组相关的dml语句组成，该组的dml语句要么全部成功，要么全部失败。

```
1 * 例如：A—B转帐，对应于如下两条sql语句
2 * update from account set money=money-100 where name='A';1000 900
3 * update from account set money=money+100 where name='B';1000 1100
4                                     2000 2000
```

* Oracle事务的相关概念

* 事务的提交：commit

* 事务的回滚：rollback

* 事务的开始

- * 连接到数据库

- * 执行DDL语句

- * 执行DML语句

- * 执行DCL语句

* 事务的结束

- * 事务COMMIT时会生成一个唯一的系统变化号（SCN）保存到事务表

- * 执行DDL语句，系统会自动执行COMMIT语句

- * 执行DCL语句，系统会自动执行COMMIT语句

- * 执行COMMIT/ROLLBACK

- * 退出/断开数据库的连接自动执行COMMIT语句

- * 进程意外终止，事务自动ROLLBACK

* 保存点（savepoint）

- * 在事务的任何地方设置保存点，以便ROLLBACK

```
1 * 演示一
2 --DROP TABLE employee;
3 --执行步骤一：创建employee表
4 CREATE TABLE employee
```

```

5 (
6     emp_no CHAR(8) PRIMARY KEY NOT NULL,    --工号, 主键, 非空
7     emp_name VARCHAR2(30) NOT NULL, --姓名, 非空
8     emp_id VARCHAR2(18), --身份证号, 代表18位整数
9     emp_age NUMBER(3,0)  --年龄
10 );
11 --执行步骤二: 插入数据
12 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441!
13 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441!
14 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441!
15 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441!
16
17 --执行步骤三: 操作employee表
18 SAVEPOINT a;
19 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张五','441!
20 SAVEPOINT b;
21 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg006','张6','4415
22 --执行步骤四: 查看employee表,
23 SELECT * FROM employee;
24 ROLLBACK TO SAVEPOINT b;
25 SELECT * FROM employee;
26 insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg007','张7','4415
27 --执行步骤五: 查看employee表,
28 SELECT * FROM employee;
29 ROLLBACK TO SAVEPOINT a;
30 SELECT * FROM employee;
31 --执行步骤六: 回滚
32 ROLLBACK;
33 --执行步骤七: 查看employee表
34 SELECT * FROM employee;
35
36 * 演示二: 测试事务自动提交
37

```

* 能够理解事务四种特性ACID

* 事务的四大特性 (ACID)

* A (Atomicity) : 原子性

* 原子性是指事务是一个不可分割的工作单位，事务中的操作要么全部成功，要么全

部失败

* C(Consistency):一致性 (能量守恒)

* 事务必须使数据库从一个一致性状态变换到另外一个一致性状态

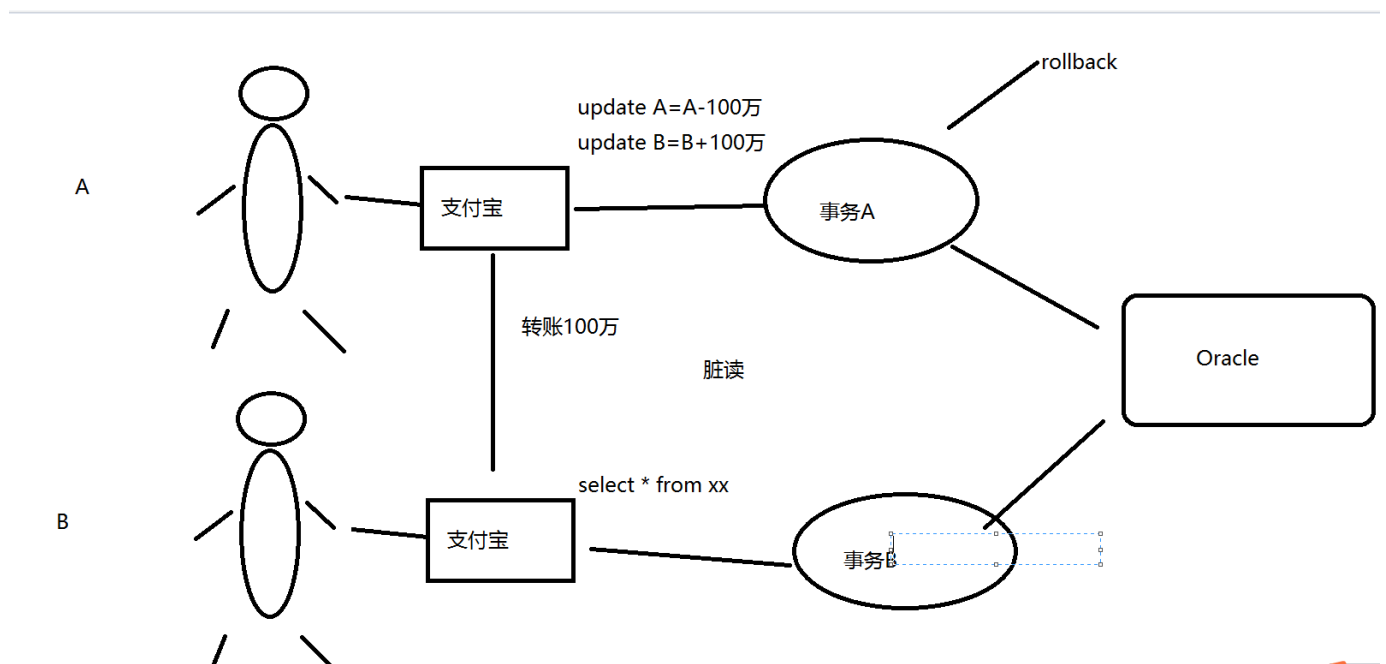
* I (Isolation) :隔离性

* 事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。

* D(durability):持久性

* 持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响

* 事务的隔离性



* 两个事务并发访问数据库数据时可能存在的问题

* 在SQL92标准中定义了4个事务的隔离级别：

- | | |
|--------------------|------------------|
| * NO_TRANSACTION | 不支持事务 |
| * READ_UNCOMMITTED | 允许脏读、不可重复读、幻读 |
| * READ_COMMITTED | 允许不可重复读、幻读，不允许脏读 |
| * REPEATABLE | 允许幻读，不允许脏读、不可重复读 |
| * SERIALIZABLE | 脏读、不可重复读、幻读都不允许 |

- * Oracle支持SQL92标准的READ_COMMITTED、SERIALIZABLE，自身特有的Read only和Read write隔离级别。

- * Oracle默认的隔离级别是READ_COMMITTED。

- * Read only：事务中不能有任何修改数据库中数据的操作语句，是Serializable的一个子集。

- * Read write：它是默认设置，该选项表示在事务中可以有访问语句、修改语句，但不经常使用。

- * Mysql支持四种隔离级别

- * Mysql默认的事务隔离级别是可重复读(Repeatable Read)

- * 假如不考虑隔离，会引发以下问题：

- * 脏读

- * 脏读指一个事务读取了另外一个事务未提交的数据。

- * 不可重复读

- * 不可重复读指在一个事务内读取表中的某一行数据，多次读取结果不同。

- * 不可重复读和脏读的区别是，脏读是读取前一事务未提交的脏数据，不可重复读是重新读取了前一事务已提交的数据。

- * 幻读（虚读）

- * 虚读(幻读)是指在一个事务内读取到了别的事务插入的数据，导致前后读取不一致。

- * MySql事务演示

- * 参考[08-MySql事务演示](#)

- * Oracle事务隔离级别演示

- * 参考[08- Oracle事务隔离级别演示](#)

- * 能够掌握使用JDBC控制数据库的事务

- * JDBC默认是提交事务

- * 常见的方法

- * Connection:setAutoCommit(false);

- * Connection:commit

- * Connection:rollback

* Connection:setSavepoint()

```
1 * 测试一
2 @Test
3     public void test1() {
4         Connection conn = null;
5         PreparedStatement psmt = null;
6         try {
7             // 1 获得Connection
8             conn = ConnectionUtils.getConnection();
9             // 开启事务
10            conn.setAutoCommit(false);
11            String sql1 = "update account set money=money-100 where name=?";
12            psmt = conn.prepareStatement(sql1);
13            psmt.setString(1, "A");
14            psmt.executeUpdate();
15            psmt.close();
16            String sql2 = "update account set money=money+100 where name=?";
17            psmt = conn.prepareStatement(sql2);
18            psmt.setString(1, "B");
19            psmt.executeUpdate();
20            // 提交事务
21            conn.commit();
22            System.out.println("成功转账");
23        } catch (Exception e) {
24        } finally {
25            ConnectionUtils.close(conn, psmt, null);
26        }
27    }
28 结果：发现A的账号少100，B账号多了100，转账成功
29
30 * 测试二
31 @Test
32     public void test2() {
33         Connection conn = null;
34         PreparedStatement psmt = null;
35         try {
36             // 1 获得Connection
37             conn = ConnectionUtils.getConnection();
```

```

38         // 开启事务
39         conn.setAutoCommit(false);
40         String sql1 = "update account set money=money-100 where name=?";
41         pstmt = conn.prepareStatement(sql1);
42         pstmt.setString(1, "A");
43         pstmt.executeUpdate();
44         pstmt.close();
45         int sum=10/0;
46         String sql2 = "update account set money=money+100 where name=?";
47         pstmt = conn.prepareStatement(sql2);
48         pstmt.setString(1, "B");
49         pstmt.executeUpdate();
50         // 提交事务
51         conn.commit();
52         System.out.println("成功转账");
53     } catch (Exception e) {
54     } finally {
55         ConnectionUtils.close(conn, pstmt, null);
56     }
57 }
58 * 结果：发现A的账号少100，B账号没有多100，不符合事务的特性
59 * 测试三
60 @Test
61 public void test3() {
62     Connection conn = null;
63     PreparedStatement pstmt = null;
64     try {
65         // 1 获得Connection
66         conn = ConnectionUtils.getConnection();
67         // 开启事务
68         conn.setAutoCommit(false);
69         String sql1 = "update account set money=money-100 where name=?";
70         pstmt = conn.prepareStatement(sql1);
71         pstmt.setString(1, "A");
72         pstmt.executeUpdate();
73         pstmt.close();
74         int sum=10/0;
75         String sql2 = "update account set money=money+100 where name=?";
76         pstmt = conn.prepareStatement(sql2);
77         pstmt.setString(1, "B");

```

```

78         psmt.executeUpdate();
79         // 提交事务
80         conn.commit();
81         System.out.println("成功转账");
82     } catch (Exception e) {
83         try {
84             conn.rollback();
85         } catch (SQLException e1) {
86             e1.printStackTrace();
87         }
88     } finally {
89         ConnectionUtils.close(conn, psmt, null);
90     }
91 }

```

结果： 当出现问题的时候：回滚了

* 测试回滚点

@Test

```

96     public void test4() {
97         Connection conn = null;
98         PreparedStatement psmt = null;
99         Savepoint sp=null;
100        try {
101            // 1 获得Connnection
102            conn = ConnectionUtils.getConnection();
103            // 开启事务
104            conn.setAutoCommit(false);
105            String sql1 = "update account set money=money-100 where name=?";
106            psmt = conn.prepareStatement(sql1);
107            psmt.setString(1, "C");
108            psmt.executeUpdate();
109            psmt.close();
110            sp = conn.setSavepoint();
111
112            int sum = 10 / 0;
113
114            String sql2 = "update account set money=money+100 where name=?";
115            psmt = conn.prepareStatement(sql2);
116            psmt.setString(1, "B");
117            psmt.executeUpdate();

```

```
118         pstmt.close();
119
120         String sql3 = "update account set money=money+100 where name=?";
121         pstmt = conn.prepareStatement(sql3);
122         pstmt.setString(1, "A");
123         pstmt.executeUpdate();
124         // 提交事务
125         conn.commit();
126         System.out.println("成功转账");
127     } catch (Exception e) {
128         try {
129             conn.rollback(sp);
130         } catch (SQLException e1) {
131             e1.printStackTrace();
132         }
133     } finally {
134         ConnectionUtils.close(conn, pstmt, null);
135     }
136 }
```

137 结果: C的账户会减少100块, 其他没有改变