* 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:setAutoCommited(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

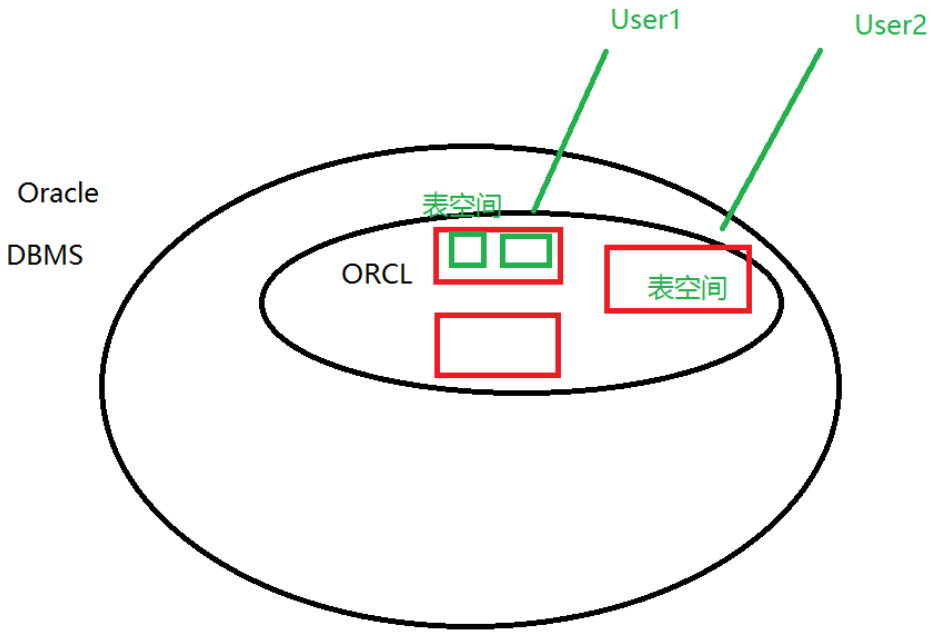
    * 表空间是数据库的逻辑划分，一个表空间只能属于一个数据库。

    * 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

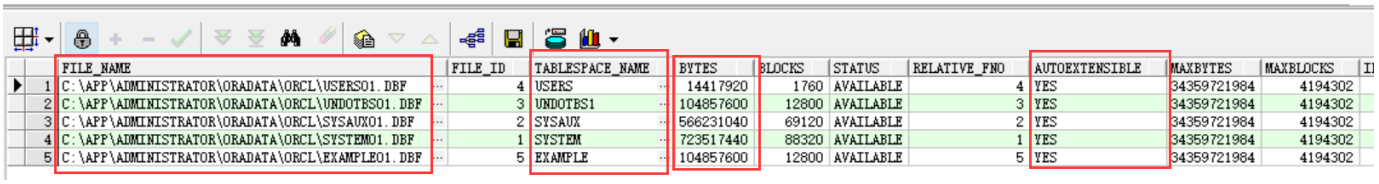
    * 永久性表空间：一般保存表、视图、过程和索引等的数据

    * 临时性表空间：只用于保存系统中短期活动的数据

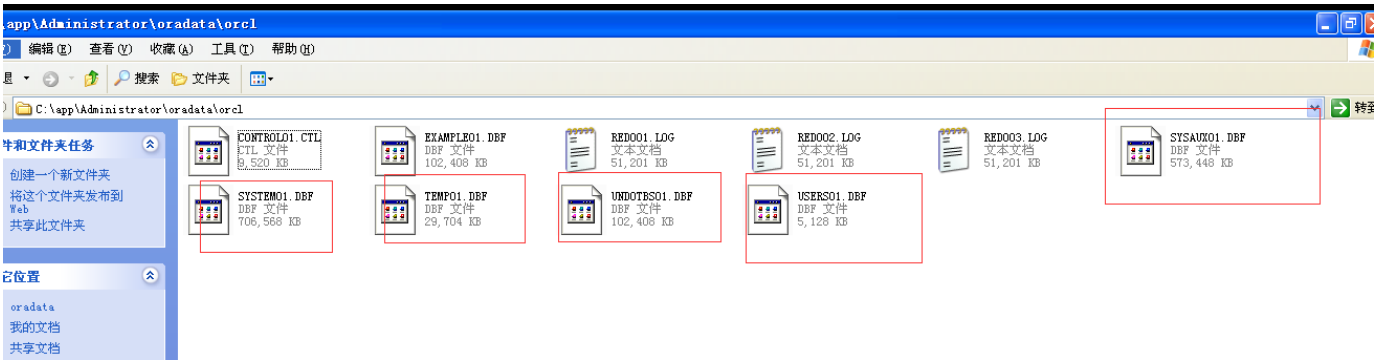
    * 撤销表空间：用来帮助回退未提交的事务数据

  * 查看表空间

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



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

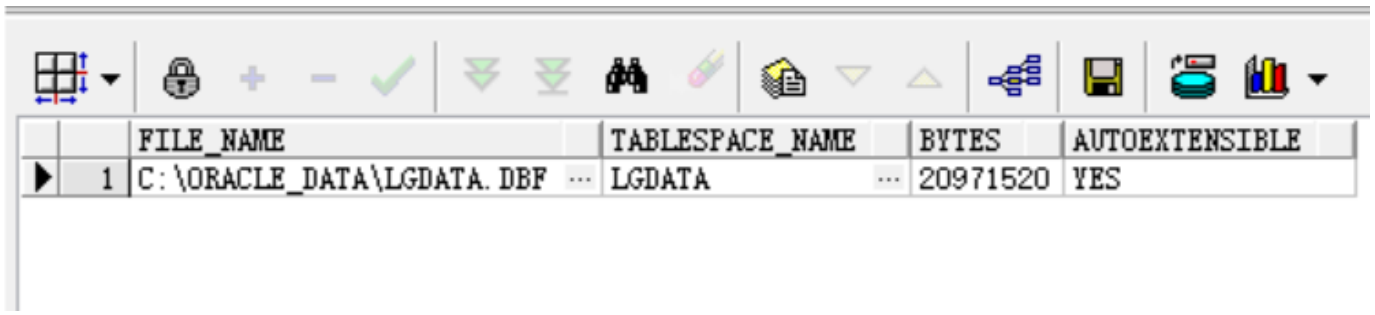


* 创建表空间

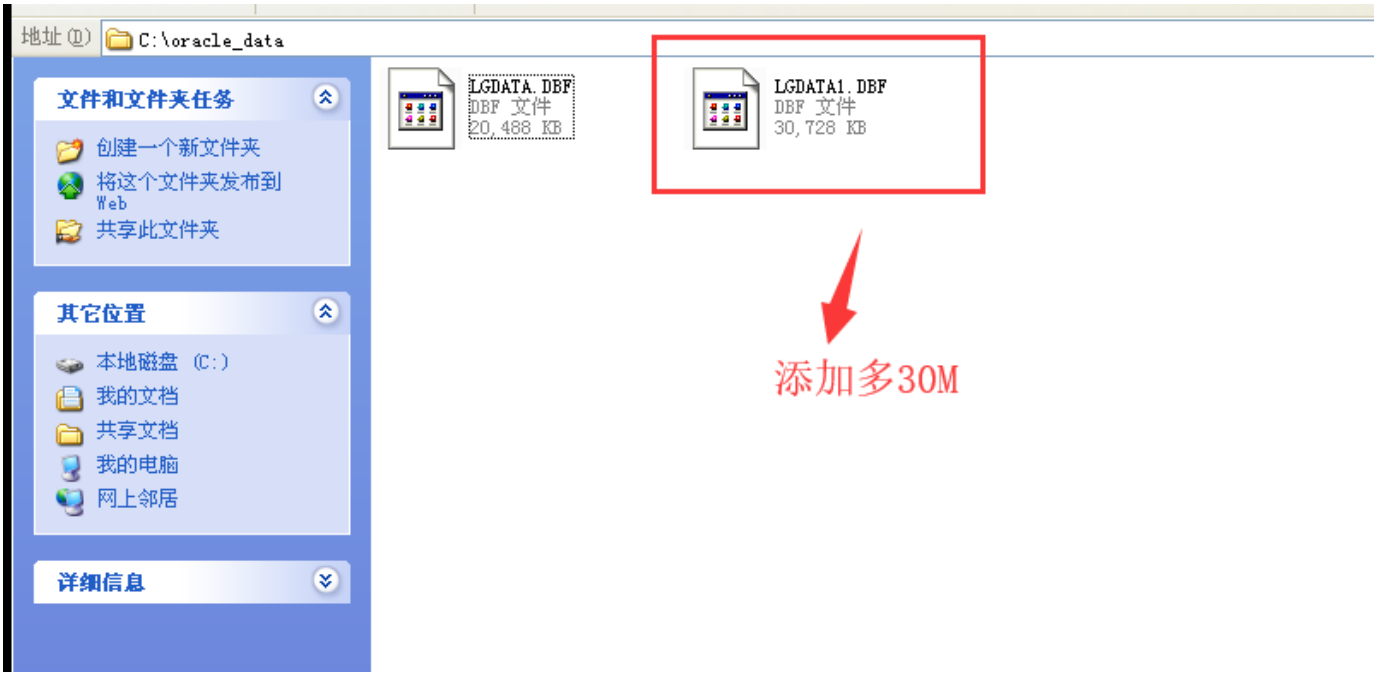
```
* 创建表空间
    * 在C盘需要先创建oracle_data目录，以存放数据文件
    * 创建表空间为lgdata，文件名字lgdata.dbf，大小为20M，可自动扩展
    * create tablespace lgdata
      datafile 'C:\oracle_data\lgdata.dbf' size 20M autoextend on;

* 查询表空间是否创建成功
  * SELECT file_name,tablespace_name,bytes,autoextensible
    FROM dba_data_files where tablespace_name='LGDATA';

* 调整表空间大小,向表空间内添加数据文件
  * alter tablespace lgdata
    add datafile 'C:\oracle_data\lgdata1.dbf' size 30M
    autoextend on;

* 删除表空间
  * drop tablespace lgdata;
```

```
*  删除表空间及数据文件
  * drop tablespace lgdata including contents and datafiles;

* 使用表空间
 * 创建表空间
  * create tablespace lgdata
    datafile 'C:\oracle_data\lgdata.dbf' size 30M
    autoextend on next 20M;

 * 创建用户使用表空间和临时
  * create user lg identified by 123123
    default tablespace lgdata
    temporary tablespace temp;

 * 查询用户的记录
  * SELECT * FROM dba_users WHERE username='LG';
 * 给用户授权
  * GRANT connect, resource TO LG;
 * 回收用户权限
  * REVOKE connect, resource FROM lg; --撤销CONNECT和RESOURCE两个角色
 * 修改用户密码
    * alter USER lg identified by 888888;
 * 删除用户（级联清除表空间的数据，但是文件还存在）
    * drop user lg cascade;

* 系统用户的介绍
  * 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。
  * 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
  * scott用户：示范用户，使用users表空间。

* 思考：
 * 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？
```

* 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。

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

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

```
(
     emp_no CHAR(8) PRIMARY KEY NOT NULL,   --工号，主键，非空
     emp_name VARCHAR2(30) NOT NULL,--姓名,非空
     emp_id VARCHAR2(18), --身份证号，代表18位整数
     emp_age NUMBER(3,0)  --年龄
);
--执行步骤二：插入数据
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg001','张大','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg002','张二','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg003','张三','441
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg004','张四','441

--执行步骤三：操作employee表
SAVEPOINT a;
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg005','张五','441
SAVEPOINT b;
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg006','张6','4415
--执行步骤四：查看employee表,
SELECT * FROM employee;
ROLLBACK TO SAVEPOINT b;
SELECT * FROM employee;
insert into employee(emp_no,emp_name,emp_id,emp_age) values('lg007','张7','4415
--执行步骤五：查看employee表,
SELECT * FROM employee;
ROLLBACK TO SAVEPOINT a;
SELECT * FROM employee;
--执行步骤六：回滚
ROLLBACK;
--执行步骤七：查看employee表
SELECT * FROM employee;

* 演示二：测试事务自动提交

```

* 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

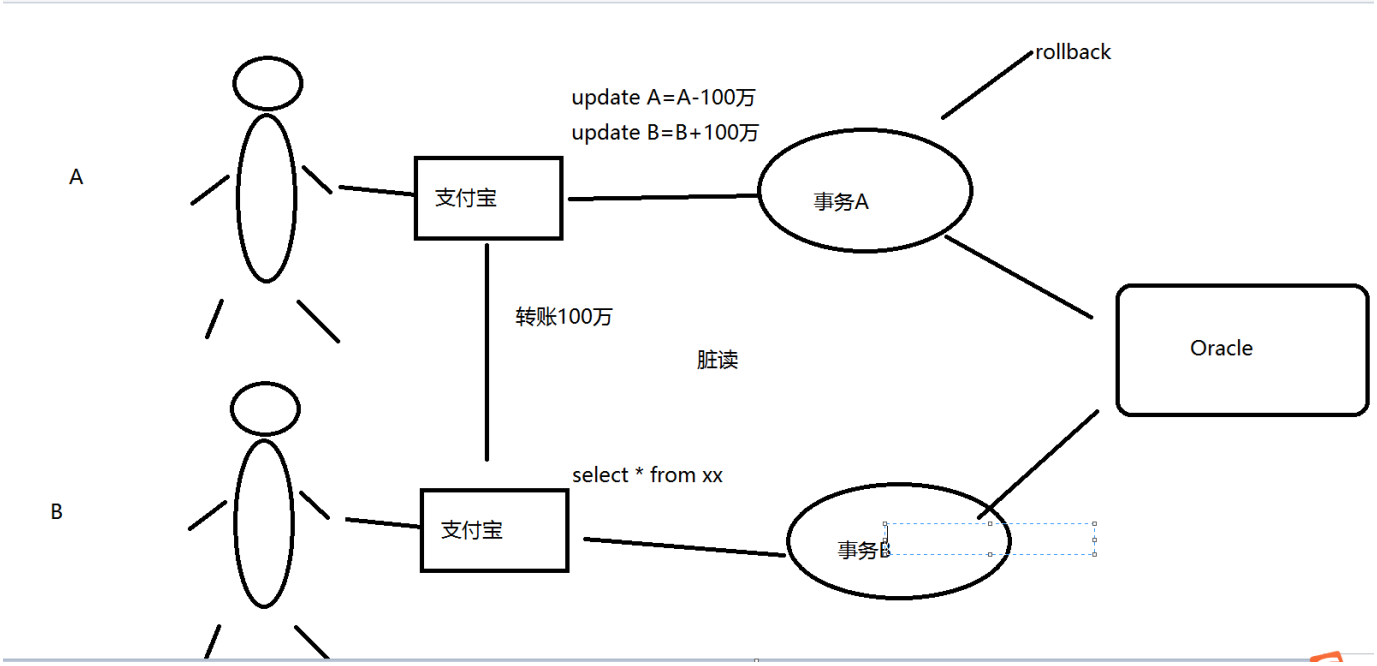
* I（Isolation）:隔离性

* 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_UNCOMMITED	允许脏读、不可重复读、幻读
* READ_COMMITED	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_COMMITED、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_COMMITED 。

* 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:setAutoCommint(false);

* Connection:commit

* Connection:rollback

* Connection:setSavepoint()

```
* 测试一
@Test
    public void test1() {
        Connection conn = null;
        PreparedStatement psmt = null;
        try {
            // 1 获得Connnection
            conn = ConnectionUtils.getConnection();
            // 开启事务
            conn.setAutoCommit(false);
            String sql1 = "update account set money=money-100 where name=?";
            psmt = conn.prepareStatement(sql1);
            psmt.setString(1, "A");
            psmt.executeUpdate();
            psmt.close();
            String sql2 = "update account set money=money+100 where name=?";
            psmt = conn.prepareStatement(sql2);
            psmt.setString(1, "B");
            psmt.executeUpdate();
            // 提交事务
            conn.commit();
            System.out.println("成功转账");
        } catch (Exception e) {
        } finally {
            ConnectionUtils.close(conn, psmt, null);
        }
    }
结果：发现A的账号少100，B账号多了100，转账成功

* 测试二
@Test
    public void test2() {
        Connection conn = null;
        PreparedStatement psmt = null;
        try {
            // 1 获得Connnection
            conn = ConnectionUtils.getConnection();
```

```
            // 开启事务
            conn.setAutoCommit(false);
            String sql1 = "update account set money=money-100 where name=?";
            psmt = conn.prepareStatement(sql1);
            psmt.setString(1, "A");
            psmt.executeUpdate();
            psmt.close();
            int sum=10/0;
            String sql2 = "update account set money=money+100 where name=?";
            psmt = conn.prepareStatement(sql2);
            psmt.setString(1, "B");
            psmt.executeUpdate();
            // 提交事务
            conn.commit();
            System.out.println("成功转账");
        } catch (Exception e) {
        } finally {
            ConnectionUtils.close(conn, psmt, null);
        }
    }
* 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
* 测试三
    @Test
    public void test3() {
        Connection conn = null;
        PreparedStatement psmt = null;
        try {
            // 1 获得Connnection
            conn = ConnectionUtils.getConnection();
            // 开启事务
            conn.setAutoCommit(false);
            String sql1 = "update account set money=money-100 where name=?";
            psmt = conn.prepareStatement(sql1);
            psmt.setString(1, "A");
            psmt.executeUpdate();
            psmt.close();
            int sum=10/0;
            String sql2 = "update account set money=money+100 where name=?";
            psmt = conn.prepareStatement(sql2);
            psmt.setString(1, "B");
```

```
            psmt.executeUpdate();
            // 提交事务
            conn.commit();
            System.out.println("成功转账");
        } catch (Exception e) {
            try {
                conn.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        } finally {
            ConnectionUtils.close(conn, psmt, null);
        }
    }
结果：  当出现问题的时候：回滚了

* 测试回滚点
@Test
    public void test4() {
        Connection conn = null;
        PreparedStatement psmt = null;
        Savepoint sp=null;
        try {
            // 1 获得Connnection
            conn = ConnectionUtils.getConnection();
            // 开启事务
            conn.setAutoCommit(false);
            String sql1 = "update account set money=money-100 where name=?";
            psmt = conn.prepareStatement(sql1);
            psmt.setString(1, "C");
            psmt.executeUpdate();
            psmt.close();
            sp = conn.setSavepoint();

            int sum = 10 / 0;

            String sql2 = "update account set money=money+100 where name=?";
            psmt = conn.prepareStatement(sql2);
            psmt.setString(1, "B");
            psmt.executeUpdate();
```

```
            psmt.close();

            String sql3 = "update account set money=money+100 where name=?";
            psmt = conn.prepareStatement(sql3);
            psmt.setString(1, "A");
            psmt.executeUpdate();
            // 提交事务
            conn.commit();
            System.out.println("成功转账");
        } catch (Exception e) {
             try {
                 conn.rollback(sp);
             } catch (SQLException e1) {
                 e1.printStackTrace();
             }
        } finally {
            ConnectionUtils.close(conn, psmt, null);
        }
    }
```

结果：C的账户会减少100块，其他没有改变