1、什么是事务Transaction

一个事务是一个完整的业务逻辑单元，不可分。

```
例如：银行账户转账，从账户A转账1000元到账户B，需要执行两条DML的update语句
update t_act set balance=balance-1000 where actno='act_A';
update t_act set balance=balance+1000 where actno='act_B';
--以上两条DML语句必须同时成功，或者同时失败，不能一条成功一条失败。要保证这样的要求需要使用数据库的"事务机制"
```

注意：跟事务相关的语句只有DML语句，即update、delete和insert。原因是DML的执行都是和数据库里的数据相关的，事务的存在就是为了保证数据的完整性、合法性以及安全性。

现实中所有的业务需求都需要事务机制，即需要多条DML语句联合完成，只需要一条是不可能的。

2、事务的特性

事务包括四大特性：ACID

A：原子性，事务是最小的工作单元，不可再分

C：一致性，事务必需保证多条DML语句同时成功或者失败

I：隔离性，事务A与事务B具有隔离，跟Java中的多线程一样

D：持久性，说的是最终数据必须持久化的保存到硬盘文件中，即commit提交，一个事务才算以成功结束

3、事务之间的隔离性

事务隔离性存在隔离级别，理论上隔离级别有四级：

```
第一级别：可读未提交（read uncommitted）
        对方事务还未提交，我们当前事务可以读取到对方未提交的数据；
        该级别会出现脏读现象（dirty read）：表示读到了脏的数据
第二级别：可读已提交（read committed）
        只有当对方事务提交数据之后，我放事务才可读到，该级别解决了脏读现象
        存在的问题是：不可重复读，我方事务先读取了硬盘文件的原始数据，但是这个事务还没结束，之后对方事务提交
        了数据，我方事务的需求还需要再一次读原先的数据，此时就无法读到原先的数据，只能读对方事务提交后的数据
        了，我方事务就没法成功结束事务。
第三级别：可重复读（repeatable read）
        解决了不可重复读的问题。永远只读我启动事务时的那份数据
        存在的问题是：读取到的数据是幻象
第四级别：序列化读/串行化读
        不允许事务并发，我这个事务开启，其他事务就没法启动
        解决了所有问题，但效率降低，需要事务排队

    Oracle的事务级别是：可读已提交
    MySQL的事务级别：可重复读
```

4、演示事务

MySQL的事务默认情况下是自动提交的（自动提交就是每执行一条DML语句后自动commit）

如何关闭自动提交，在每次开始事务前执行（start transaction;）。

```
--演示
--1、建表
drop table if exists t_user2;
create table t_user2(
    id int primary key auto_increment,
    username varchar(255)
);

--2、演示MySQL自动commit
mysql> insert into t_user2(username) values('zhangsan');
Query OK, 1 row affected (1.89 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
+----+----------+
|  1 | zhangsan |
+----+----------+
1 row in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
+----+----------+
|  1 | zhangsan |
+----+----------+
1 row in set (0.00 sec)
--通过上面操作可以看出rollback回滚失败，自动提交了

--3、演示每次事务前关闭自动提交机制（注意每次事务都要关才行）
mysql> start transaction;--关掉一次只针对一次事务
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t_user2(username) values('lisi');
Query OK, 1 row affected (0.00 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
+----+----------+
|  1 | zhangsan |
|  2 | lisi     |
+----+----------+
2 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.13 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
```

```
+----+----------+
|  1 | zhangsan |
+----+----------+
1 row in set (0.00 sec)
-------------------------------------------------------------
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t_user2(username) values('lisi');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t_user2(username) values('wangwu');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t_user2(username) values('zhaoliu');
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.13 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
+----+----------+
|  1 | zhangsan |
|  3 | lisi     |
|  4 | wangwu   |
|  5 | zhaoliu  |
+----+----------+
4 rows in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t_user2;
+----+----------+
| id | username |
+----+----------+
|  1 | zhangsan |
|  3 | lisi     |
|  4 | wangwu   |
|  5 | zhaoliu  |
+----+----------+
4 rows in set (0.00 sec)
--每次关闭自动提交后只要执行了commit或rollback说明一个事务结束了，下一次在执行DML语句即开始事务不希望自动提交还得再关掉这个机制，可以看到id主键字段即使原先的数据没了，用过的主键值不再重复出现，但这没有影响的，不用纠结
```

5、隔离级别演示

设置事务的全局隔离级别：
set global transaction isolation level 隔离级别（包括read uncommitted、read committed、repeatable read和serializable）
查看事务的全局隔离级别：

```
mysql> select @@global.tx_isolation;
+-----------------------+
| @@global.tx_isolation |
+-----------------------+
| REPEATABLE-READ       |
+-----------------------+
1 row in set, 1 warning (0.00 sec)
```