事务

1.1 什么是事务?

一个事务其实就是一个完整的业务逻辑。
是一个最小的工作单元。不可再分。

```
什么是一个完整的业务逻辑?
    假设转账，从A账户向B账户中转账10000.
    将A账户的钱减去10000（update语句）
    将B账户的钱加上10000（update语句）
    这就是一个完整的业务逻辑。

    以上的操作是一个最小的工作单元，要么同时成功，要么同时失败，不可再分。
    这两个update语句要求必须同时成功或者同时失败，这样才能保证钱是正确的。
```

1.2 只有DML语句才会有事务这一说，其它语句和事务无关！！！

insert
delete
update
只有以上的三个语句和事务有关系，其它都没有关系。

```
因为 只有以上的三个语句是数据库表中数据进行增、删、改的。
只要你的操作一旦涉及到数据的增、删、改，那么就一定要考虑安全问题。

数据安全第一位！！！
```

1.3 假设所有的业务，只要一条DML语句就能完成，还有必要存在事务机制吗?

正是因为做某件事的时候，需要多条DML语句共同联合起来才能完成，
所以需要事务的存在。如果任何一件复杂的事儿都能一条DML语句搞定，
那么事务则没有存在的价值了。

到底什么是事务呢?
说到底，说到本质上，一个事务其实就是多条DML语句同时成功，或者同时失败!

事务：就是批量的DML语句同时成功，或者同时失败!

一个事务是一个DML语句的集合。这是一个充分不必要条件。一个DML语句的集合不一定一个事务。

1.4 事务是怎么做到多条DML语句同时成功和同时失败的呢?

```
InnoDB存储引擎：提供一组用来记录事务性活动的日志文件

事务开启了：
insert
insert
insert
delete
update
```

```
update
update
事务结束了!

在事务的执行过程中，每一条DML的操作都会记录到“事务性活动的日志文件”中。
在事务的执行过程中，我们可以提交事务，也可以回滚事务。

提交事务？
    清空事务性活动的日志文件，将数据全部彻底持久化到数据库表中。
    提交事务标志着，事务的结束。并且是一种全部成功的结束。

回滚事务？
    将之前所有的DML操作全部撤销，并且清空事务性活动的日志文件
    回滚事务标志着，事务的结束。并且是一种全部失败的结束。
```

1.5 怎么提交事务，怎么回滚事务?

提交事务：commit; 语句
回滚事务：rollback; 语句（回滚永远都是只能回滚到上一次的提交点！）

```
事务对应的英语单词是：transaction

测试一下，在mysql当中默认的事务行为是怎样的？
    mysql默认情况下是支持自动提交事务的。（自动提交）
    什么是自动提交？
        每执行一条DML语句，则提交一次！

    这种自动提交实际上是不符合我们的开发习惯，因为一个业务
    通常是需要多条DML语句共同执行才能完成的，为了保证数据
    的安全，必须要求同时成功之后再提交，所以不能执行一条
    就提交一条。

怎么将mysql的自动提交机制关闭掉呢？
    先执行这个命令：start transaction;
```

演示事务：

回滚事务：

```
    mysql> use bjpowernode;
    Database changed
    mysql> select * from dept_bak;
    Empty set (0.00 sec)

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

    mysql> insert into dept_bak values(10,'abc', 'tj');
    Query OK, 1 row affected (0.00 sec)

    mysql> insert into dept_bak values(10,'abc', 'tj');
    Query OK, 1 row affected (0.00 sec)

    mysql> select * from dept_bak;
    +--------+-------+------+
    | DEPTNO | DNAME | LOC  |
```

```
+--------+-------+------+
|     10 | abc   | tj   |
|     10 | abc   | tj   |
+--------+-------+------+
2 rows in set (0.00 sec)

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

mysql> select * from dept_bak;
Empty set (0.00 sec)
```

提交事务：

mysql> use bjpowernode;
Database changed
mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC |
+--------+-------+------+
| 10 | abc | bj |
+--------+-------+------+
1 row in set (0.00 sec)

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

mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

```
mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

mysql> insert into dept_bak values(20,'abc
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | bj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
+--------+-------+------+
4 rows in set (0.00 sec)

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

```
mysql> select * from dept_bak;
+--------+-------+------+
| DEPTNO | DNAME | LOC  |
+--------+-------+------+
|     10 | abc   | bj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
|     20 | abc   | tj   |
+--------+-------+------+
4 rows in set (0.00 sec)
```

1.6 事务包括4个特性?

```
A：原子性
    说明事务是最小的工作单元。不可再分。

C：一致性
    所有事务要求，在同一个事务当中，所有操作必须同时成功，或者同时失败，
    以保证数据的一致性。

I：隔离性
    A事务和B事务之间具有一定的隔离。
    教室A和教室B之间有一道墙，这道墙就是隔离性。
    A事务在操作一张表的时候，另一个事务B也操作这张表会那样？？？

D：持久性
    事务最终结束的一个保障。事务提交，就相当于将没有保存到硬盘上的数据
    保存到硬盘上！
```

1.7 重点研究一下事务的隔离性！！！

隔离性是为了解决多事务并发时的问题（比如多事务并发时，多个事务同时操作同一个数据，此时就会产生问题），多事务并发就是多线程并发，多个线程上都执行着事务。

```
A教室和B教室中间有一道墙，这道墙可以很厚，也可以很薄。这就是事务的隔离级别。
这道墙越厚，表示隔离级别就越高。

事务和事务之间的隔离级别有哪些呢？4个级别

    读未提交：read uncommitted（最低的隔离级别）《没有提交就读到了》
        什么是读未提交？
            事务A可以读取到事务B未提交的数据。
        这种隔离级别存在的问题就是：
            脏读现象！(Dirty Read)
            我们称读到了脏数据。
        这种隔离级别一般都是理论上的，大多数的数据库隔离级别都是二档起步！

    读已提交：read committed《提交之后才能读到》
        什么是读已提交？
            事务A只能读取到事务B提交之后的数据。
        这种隔离级别解决了什么问题？
            解决了脏读的现象。
        这种隔离级别存在什么问题？
            不可重复读取数据。
            什么是不可重复读取数据呢？
                在事务开启之后，第一次读到的数据是3条，当前事务还没有
```

结束，可能第二次再读取的时候，读到的数据是4条，3不等于4
称为不可重复读取。

这种隔离级别是比较真实的数据，每一次读到的数据是绝对的真实。
oracle数据库默认的隔离级别是：read committed

可重复读：repeatable read《提交之后也读不到，永远读取的都是刚开启事务时的数据》
什么是可重复读取？
事务A开启之后，不管是多久，每一次在事务A中读取到的数据
都是一致的。即使事务B将数据已经修改，并且提交了，事务A
读取到的数据还是没有发生改变，这就是可重复读。
可重复读解决了什么问题？
解决了不可重复读取数据。
可重复读存在的问题是什么？
可以会出现幻影读。
每一次读取到的数据都是幻象。不够真实！

早晨9点开始开启了事务，只要事务不结束，到晚上9点，读到的数据还是那样！
读到的是假象。不够绝对的真实。

mysql中默认的事务隔离级别就是这个！！！！！！！！！！！

序列化/串行化：serializable（最高的隔离级别）
这是最高隔离级别，效率最低。解决了所有的问题。
这种隔离级别表示事务排队，不能并发！
synchronized，线程同步（事务同步）
每一次读取到的数据都是最真实的，并且效率是最低的。

1.8 验证各种隔离级别

查看隔离级别：SELECT @@tx_isolation

```
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+
```

这是mysql默认的隔离级别