本篇讲诉数据库中事务的四大特性（ACID），并且将会详细地说明事务的隔离级别。

如果一个数据库声称支持事务的操作，那么该数据库必须要具备以下四个特性：

⑴ 原子性（Atomicity）

原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，这和前面两篇博客介绍事务的功能是一样的概念，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

⑵ 一致性（Consistency）

一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。

拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。

⑶ 隔离性（Isolation）

隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

关于事务的隔离性数据库提供了多种隔离级别，稍后会介绍到。

⑷ 持久性（Durability）

持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。

以上介绍完事务的四大特性(简称ACID)，现在重点来说明下事务的隔离性，当多个线程都开启事务操作数据库中的数据时，数据库系统要能进行隔离操作，以保证

各个线程获取数据的准确性，在介绍数据库提供的各种隔离级别之前，我们先看看如果不考虑事务的隔离性，会发生的几种问题：

1，脏读

脏读是指在一个事务处理过程里读取了另一个未提交的事务中的数据。

当一个事务正在多次修改某个数据，而在这个事务中这多次的修改都还未提交，这时一个并发的事务来访问该数据，就会造成两个事务得到的数据不一致。例如：用户A向用户B转账100元，对应SQL命令如下

```
update account set money=money+100 where name='B';  (此时A通知B)

update account set money=money - 100 where name='A';
```

当只执行第一条SQL时，A通知B查看账户，B发现确实钱已到账（此时即发生了脏读），而之后无论第二条SQL是否执行，只要该事务不提交，则所有操作都将回滚，那么当B以后再次查看账户时就会发现钱其实并没有转。

2，不可重复读

不可重复读是指在对于数据库中的某个数据，一个事务范围内多次查询却返回了不同的数据值，这是由于在查询间隔，被另一个事务修改并提交了。

例如事务T1在读取某一数据，而事务T2立马修改了这个数据并且提交事务给数据库，事务T1再次读取该数据就得到了不同的结果，发送了不可重复读。

不可重复读和脏读的区别是，脏读是某一事务读取了另一个事务未提交的脏数据，而不可重复读则是读取了前一事务提交的数据。

在某些情况下，不可重复读并不是问题，比如我们多次查询某个数据当然以最后查询得到的结果为主。但在另一些情况下就有可能发生问题，例如对于同一个数据A和B依次查询就可能不同，A和B就可能打起来了……

3，虚读(幻读)

幻读是事务非独立执行时发生的一种现象。例如事务T1对一个表中所有的行的某个数据项做了从“1”修改为“2”的操作，这时事务T2又对这个表中插入了一行数据项，而这个数据项的数值还是为“1”并且提交给数据库。而操作事务T1的用户如果再查看刚刚修改的数据，会发现还有一行没有修改，其实这行是从事务T2中添加的，就好像产生幻觉一样，这就是发生了幻读。

幻读和不可重复读都是读取了另一条已经提交的事务（这点就脏读不同），所不同的是不可重复读查询的都是同一个数据项，而幻读针对的是一批数据整体（比如数据的个数）。

现在来看看MySQL数据库为我们提供的四种隔离级别：

① Serializable (串行化)：可避免脏读、不可重复读、幻读的发生。

② Repeatable read (可重复读)：可避免脏读、不可重复读的发生。

③ Read committed (读已提交)：可避免脏读的发生。

④ Read uncommitted (读未提交)：最低级别，任何情况都无法保证。

以上四种隔离级别最高的是Serializable级别，最低的是Read uncommitted级别，当然级别越高，执行效率就越低。像Serializable这样的级别，就是以锁表的方式(类似于Java多线程中的锁)使得其他的线程只能在锁外等待，所以平时选用何种隔离级别应该根据实际情况。在MySQL数据库中默认的隔离级别为Repeatable read (可重复读)。

在MySQL数据库中，支持上面四种隔离级别，默认的为Repeatable read (可重复读)；而在Oracle数据库中，只支持Serializable (串行化)级别和Read committed (读已提交)这两种级别，其中默认的为Read committed级别。

在MySQL数据库中查看当前事务的隔离级别：

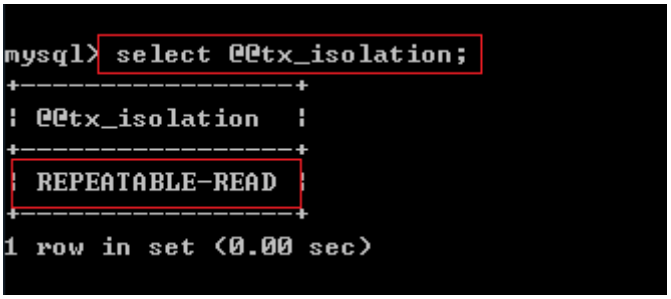
```
select @@tx_isolation;
```

在MySQL数据库中设置事务的隔离 级别：

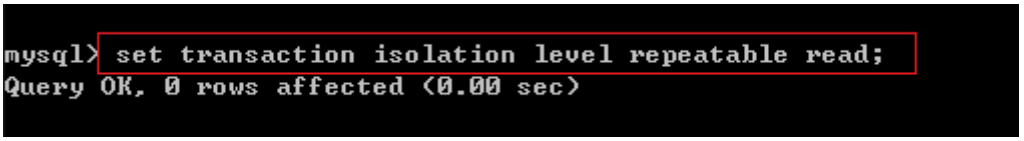
```
set  [glogal | session]  transaction isolation level 隔离级别名称;

set tx_isolation=’隔离级别名称;’
```

例1：查看当前事务的隔离级别：



例2：将事务的隔离级别设置为Read uncommitted级别：



或：

```
mysql> set tx_isolation='read-uncommitted';
Query OK, 0 rows affected (0.00 sec)

mysql> select @@tx_isolation;
+------------------+
| @@tx_isolation   |
+------------------+
| READ-UNCOMMITTED |
+------------------+
1 row in set (0.00 sec)
```

记住：设置数据库的隔离级别一定要是在开启事务之前！

如果是使用JDBC对数据库的事务设置隔离级别的话，也应该是在调用Connection对象的setAutoCommit(false)方法之前。调用Connection对象的setTransactionIsolation(level)即可设置当前链接的隔离级别，至于参数level，可以使用Connection对象的字段：

字段摘要	
static int	**TRANSACTION_NONE** 指示事务不受支持的常量。
static int	**TRANSACTION_READ_COMMITTED** 指示不可以发生脏读的常量；不可重复读和虚读可以发生。
static int	**TRANSACTION_READ_UNCOMMITTED** 指示可以发生脏读 (dirty read)、不可重复读和虚读 (phantom read) 的常量。
static int	**TRANSACTION_REPEATABLE_READ** 指示不可以发生脏读和不可重复读的常量；虚读可以发生。
static int	**TRANSACTION_SERIALIZABLE** 指示不可以发生脏读、不可重复读和虚读的常量。

在JDBC中设置隔离级别的部分代码：

```
Connection conn = null;
Statement st = null;
ResultSet rs = null;

try{
    conn = JdbcUtils.getConnection();
    //设置该链接的隔离级别
    conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
    conn.setAutoCommit(false);  //开启事务
```

后记：隔离级别的设置只对当前链接有效。对于使用MySQL命令窗口而言，一个窗口就相当于一个链接，当前窗口设置的隔离级别只对当前窗口中的事务有效；对于JDBC操作数据库来说，一个Connection对象相当于一个链接，而对于Connection对象设置的隔离级别只对该Connection对象有效，与其他链接Connection对象无关。