[**JDBC处理事务**](http://www.cnblogs.com/gdwkong/p/7633016.html)

<https://www.cnblogs.com/gdwkong/p/7633016.html>

一、什么是事务？

　　在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

二、事务是必须满足4个条件（ACID）

* **事务的原子性（ Atomicity）：**一组事务，要么成功；要么撤回。
* **一致性 （Consistency）：**事务执行后，数据库状态与其他业务规则保持一致。如转账业务，无论事务执行成功否，参与转账的两个账号余额之和应该是不变的。
* **隔离性（Isolation）：**事务独立运行。一个事务处理后的结果，影响了其他事务，那么其他事务会撤回。事务的100%隔离，需要牺牲速度。
* **持久性（Durability）：**软、硬件崩溃后，InnoDB数据表驱动会利用日志文件重构修改。可靠性和高速度不可兼得， innodb\_flush\_log\_at\_trx\_commit 选项 决定什么时候吧事务保存到日志里。

三、MySQL中的事务

    在默认情况下，MySQL每执行一条SQL语句，都是一个单独的事务。如果需要在一个事务中包含多条SQL语句，那么需要开启事务和结束事务。

* 开启事务：start transaction
* 结束事务：commit或rollback

     在执行SQL语句之前，先执行start transaction，这就开启了一个事务（事务的起点），然后可以去执行多条SQL语句，最后要结束事务，commit表示提交，即事务中的多条SQL语句所作出的影响会持久到数据库中，或者rollback，表示回滚到事务的起点，之前做的所有操作都被撤销了。

[复制代码](javascript:void(0);)

1 mysql> SELECT \* FROM account;

2 +----+------+---------+

3 | id | NAME | balance |

4 +----+------+---------+

5 | 1 | zs | 1000.00 |

6 | 2 | ls | 1000.00 |

7 | 3 | ww | 1000.00 |

8 +----+------+---------+

9 3 rows in set (0.00 sec)

10

11 mysql> START TRANSACTION;

12 Query OK, 0 rows affected (0.00 sec)

13

14 mysql> UPDATE account SET balance=900 WHERE name = 'zs';

15 Query OK, 1 row affected (0.00 sec)

16 Rows matched: 1 Changed: 1 Warnings: 0

17

18 mysql> SELECT \* FROM account;

19 +----+------+---------+

20 | id | NAME | balance |

21 +----+------+---------+

22 | 1 | zs | 900.00 |

23 | 2 | ls | 1000.00 |

24 | 3 | ww | 1000.00 |

25 +----+------+---------+

26 3 rows in set (0.00 sec)

27

28 mysql> UPDATE account SET balance=1100 WHERE name = 'ls';

29 Query OK, 1 row affected (0.00 sec)

30 Rows matched: 1 Changed: 1 Warnings: 0

31

32 mysql> SELECT \* FROM account;

33 +----+------+---------+

34 | id | NAME | balance |

35 +----+------+---------+

36 | 1 | zs | 900.00 |

37 | 2 | ls | 1100.00 |

38 | 3 | ww | 1000.00 |

39 +----+------+---------+

40 3 rows in set (0.00 sec)

41

42 mysql> ROLLBACK;

43 Query OK, 0 rows affected (0.00 sec)

44

45 mysql> SELECT \* FROM account;

46 +----+------+---------+

47 | id | NAME | balance |

48 +----+------+---------+

49 | 1 | zs | 1000.00 |

50 | 2 | ls | 1000.00 |

51 | 3 | ww | 1000.00 |

52 +----+------+---------+

53 3 rows in set (0.00 sec)

54

55 mysql> START TRANSACTION;

56 Query OK, 0 rows affected (0.00 sec)

57

58 mysql> UPDATE account SET balance=balance-100 WHERE name = 'zs';

59 Query OK, 1 row affected (0.00 sec)

60 Rows matched: 1 Changed: 1 Warnings: 0

61

62 mysql> SELECT \* FROM account;

63 +----+------+---------+

64 | id | NAME | balance |

65 +----+------+---------+

66 | 1 | zs | 900.00 |

67 | 2 | ls | 1000.00 |

68 | 3 | ww | 1000.00 |

69 +----+------+---------+

70 3 rows in set (0.00 sec)

71

72 mysql> UPDATE account SET balance=balance+100 WHERE name = 'ls';

73 Query OK, 1 row affected (0.00 sec)

74 Rows matched: 1 Changed: 1 Warnings: 0

75

76 mysql> SELECT \* FROM account;

77 +----+------+---------+

78 | id | NAME | balance |

79 +----+------+---------+

80 | 1 | zs | 900.00 |

81 | 2 | ls | 1100.00 |

82 | 3 | ww | 1000.00 |

83 +----+------+---------+

84 3 rows in set (0.00 sec)

85

86 mysql> commit;

87 Query OK, 0 rows affected (0.02 sec)

88

89 mysql> SELECT \* FROM account;

90 +----+------+---------+

91 | id | NAME | balance |

92 +----+------+---------+

93 | 1 | zs | 900.00 |

94 | 2 | ls | 1100.00 |

95 | 3 | ww | 1000.00 |

96 +----+------+---------+

97 3 rows in set (0.00 sec)

[复制代码](javascript:void(0);)

四、JDBC事务

在JDBC中处理事务，都是通过Connection完成的。

**同一事务中所有的操作，都在使用同一个Connection对象。**

①JDBC中的事务

Connection的三个方法与事务有关：

* setAutoCommit（boolean）:设置是否为自动提交事务，如果true（默认值为true）表示自动提交，也就是每条执行的SQL语句都是一个单独的事务，如果设置为false，那么相当于开启了事务了；**con.setAutoCommit(false) 表示开启事务。**
* **commit（）：提交结束事务。**
* **rollback（）：回滚结束事务。**

**JDBC处理事务的代码格式：**

[复制代码](javascript:void(0);)

try{

con.setAutoCommit(false);//开启事务

......

con.commit();//try的最后提交事务

} catch（） {

con.rollback();//回滚事务

}

[复制代码](javascript:void(0);)

示例：

[复制代码](javascript:void(0);)

1 public class AccountDao {

2 /\*

3 \* 修改指定用户的余额

4 \* \*/

5 public void updateBalance(Connection con, String name,double balance) {

6 try {

7 String sql = "UPDATE account SET balance=balance+? WHERE name=?";

8 PreparedStatement pstmt = con.prepareStatement(sql);

9 pstmt.setDouble(1,balance);

10 pstmt.setString(2,name);

11 pstmt.executeUpdate();

12 }catch (Exception e) {

13 throw new RuntimeException(e);

14 }

15 }

16 }

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

1 import cn.itcast.jdbc.JdbcUtils;

2 import org.junit.Test;

3 import java.sql.Connection;

4 import java.sql.SQLException;

5

6 public class Demo1 {

7 /\*

8 \* 演示转账方法

9 \* 所有对Connect的操作都在Service层进行的处理

10 \* 把所有connection的操作隐藏起来，这需要使用自定义的小工具（day19\_1）

11 \* \*/

12 public void transferAccounts(String from,String to,double money) {

13 //对事务的操作

14 Connection con = null;

15 try{

16 con = JdbcUtils.getConnection();

17 con.setAutoCommit(false);

18 AccountDao dao = new AccountDao();

19 dao.updateBalance(con,from,-money);//给from减去相应金额

20 if (true){

21 throw new RuntimeException("不好意思，转账失败");

22 }

23 dao.updateBalance(con,to,+money);//给to加上相应金额

24 //提交事务

25 con.commit();

26

27 } catch (Exception e) {

28 try {

29 con.rollback();

30 } catch (SQLException e1) {

31 e.printStackTrace();

32 }

33 throw new RuntimeException(e);

34 }

35 }

36 @Test

37 public void fun1() {

38 transferAccounts("zs","ls",100);

39 }

40 }

[复制代码](javascript:void(0);)

五、事务隔离级别

1、事务的并发读问题

* 脏读：读取到另外一个事务未提交数据（不允许出来的事）；
* 不可重复读：两次读取不一致；
* 幻读（虚读）：读到另一事务已提交数据。

2、并发事务问题

因为并发事务导致的问题大致有5类，其中两类是更新问题三类是读问题。

* **脏读（dirty read）：读到另一个事务的未提交新数据，即读取到了脏数据；**
* **不可重复读（unrepeatable）：对同一记录的两次读取不一致，因为另一事务对该记录做了修改；**
* **幻读（虚读）（phantom read）：对同一张表的两次查询不一致，因为另一事务插入了一条记录。**

3、四大隔离级别

    4个等级的事务隔离级别，在相同的数据环境下，使用相同的输入，执行相同的工作，根据不同的隔离级别，可以导致不同的结果。不同事务隔离级别能够解决的数据并发问题的能力是不同的。

1、SERIALIZABLE(串行化)

* 不会出现任何并发问题，因为它是对同一数据的访问是串行的，非并发访问的；
* 性能最差

**2、REPEATABLE READ(可重复读)（MySQL）**

* **防止脏读和不可重复读，不能处理幻读**
* **性能比SERIALIZABLE好**

**3、READ COMMITTED(读已提交数据)（Oracle）**

* **防止脏读，不能处理不可重复读和幻读；**
* **性能比REPEATABLE READ好**

4、READ UNCOMMITTED(读未提交数据)

* 可能出现任何事物并发问题，什么都不处理。
* 性能最好

六、MySQL隔离级别

MySQL的默认隔离级别为Repeatable read,可以通过下面语句查看：

SELECT @@`TX\_ISOLATION`;

也可以通过下面语句来设置当前连接的隔离级别：

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ ;//[4选1]

七、JDBC设置隔离级别

con.setTransactionIsolation(int level) :参数可选值如下：

* Connection.TRANSACTION\_READ\_UNCOMMITTED；
* Connection.TRANSACTION\_READ\_COMMITTED；
* Connection.TRANSACTION\_REPEATABLE\_READ；
* Connection.TRANSACTION\_READ\_SERIALIZABLE。