**面试：Mybatis事务请讲解一下？**

<https://mp.weixin.qq.com/s/BmCoNKnbxEDHt2-mHDid8A>

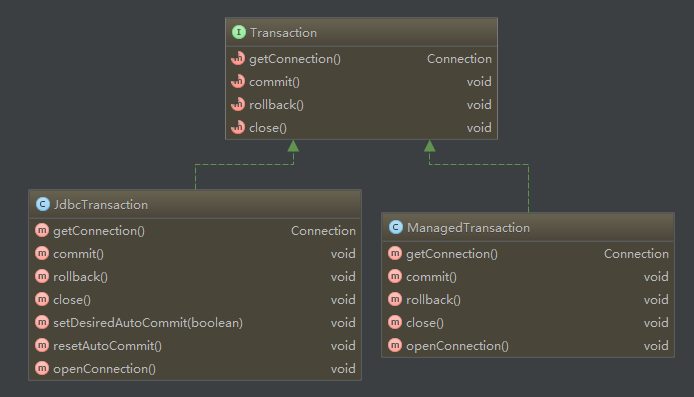
因此，记住事务的三个真实存在的方法，不要被各种事务状态名词所迷惑，它们分别是：**conn.setAutoCommit()**、**conn.commit()**、**conn.rollback()**。conn.close()含义为关闭一个数据库连接，这已经不再是事务方法了。

**1. Mybaits中的事务接口Transaction**

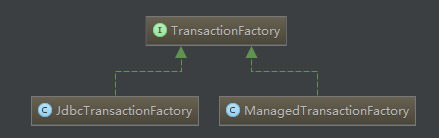
public interface Transaction {  
    Connection getConnection() throws SQLException;  
    void commit() throws SQLException;  
    void rollback() throws SQLException;  
    void close() throws SQLException;  
}

有了文章开头的分析，当你再次看到close()方法时，千万别再认为是关闭一个事务了，而是关闭一个conn连接，或者是把conn连接放回连接池内。

事务类层次结构图：



## 2. 事务工厂TransactionFactory



顾名思义，一个生产JdbcTransaction实例，一个生产ManagedTransaction实例。两个毫无实际意义的工厂类，除了new之外，没有其他代码。

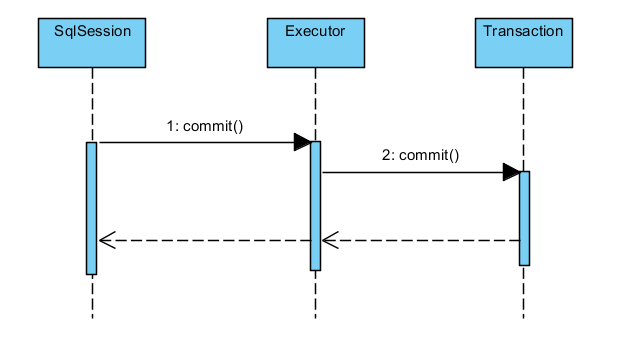
<transactionManager type="JDBC" />

mybatis-config.xml配置文件内，可配置事务管理类型。

## 3. Transaction的用法

无论是SqlSession，还是Executor，它们的事务方法，最终都指向了Transaction的事务方法，即都是由Transaction来完成事务提交、回滚的。

配一个简单的时序图。





## 4. 你可能关心的有关事务的几种特殊场景表现（重要）

### 1. 一个conn生命周期内，可以存在无数多个事务。

### 2. autoCommit=false，没有执行commit()，仅执行close()，会发生什么？

**因此，得出结论：autoCommit=false，但是没有手动commit，在sqlSession.close()时，Mybatis会将事务进行rollback()操作，然后才执行conn.close()关闭连接，当然数据最终也就没能持久化到数据库中了。**

### 3. autoCommit=false，没有commit，也没有close，会发生什么？

即不commit，也不close。

**结论：insert后，jvm结束前，如果事务隔离级别是read uncommitted，我们可以查到该条记录。jvm结束后，事务被rollback()，记录消失。通过断点debug方式，你可以看到效果。**

**这说明JDBC驱动实现，已经考虑到这样的特例情况，底层已经有相应的处理机制了。这也超出了我们的探究范围。**

**注：无参的openSession()方法，会自动设置autoCommit=false。**

总结：Mybatis的JdbcTransaction，和纯粹的Jdbc事务，几乎没有差别，它仅是扩展支持了连接池的connection。另外，需要明确，无论你是否手动处理了事务，只要是对数据库进行任何update操作（update、delete、insert），都一定是在事务中进行的，这是数据库的设计规范之一。