用 STM 实现转账

我们曾经在《05 | 一不小心就死锁了，怎么办？》这篇文章中，讲到了并发转账的例子，示例代码如下。简单地使用 synchronized 将 transfer() 方法变成同步方法并不能解决并发问题，因为还存在死锁问题。

复制代码

```
class UnsafeAccount {
  // 余额
  private long balance;
  // 构造函数
  public UnsafeAccount(long balance) {
    this.balance = balance;
  }
  // 转账
  void transfer(UnsafeAccount target, long amt){
    if (this.balance > amt) {
      this.balance -= amt;
      target.balance += amt;
    }
  }
}

```

该转账操作若使用数据库事务就会非常简单，如下面的示例代码所示。如果所有 SQL 都正常执行，则通过 commit() 方法提交事务；如果 SQL 在执行过程中有异常，则通过 rollback() 方法回滚事务。数据库保证在并发情况下不会有死锁，而且还能保证前面我们说的原子性、一致性、隔离性和持久性，也就是 ACID。

复制代码

```
Connection conn = null;
try{
  // 获取数据库连接
  conn = DriverManager.getConnection();
  // 设置手动提交事务
  conn.setAutoCommit(false);
  // 执行转账 SQL
  ......
  // 提交事务
  conn.commit();
```