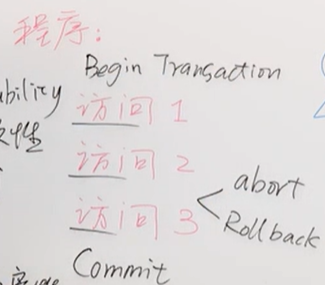
* 事务处理的概念
  1. 保证数据正确性
  2. 对数据库，一个事务是一个程序
  3. 保证事务操作级别的原子性
* 事务的功能和性质 ACID
  1. 性质
     + A原子性：事务或者没有做过，或者全部完成
     + C一致性：数据一致
     + I隔离性：多个事务同时运行是隔离的
       - Read Committed
       - Repeatable Read
     + D持久性：一旦提交无法撤销
  2. 机制：
     + 日志：AD
     + 并发控制：I  
       
* 合理使用事务
  1. 原则：事务应该是短小的
  2. 订电影票流程
     + 获得空闲座位
     + 展示座位
     + 用户选择座位
     + 用户提交订票请求  
       Begin Transection
     + 判断座位是否仍空闲
     + IF 空闲 Than {分配给用户 Commit；}  
       Else {abort；Info；}
  3. 购买商品的过程  
     Begin Transection
     + IF 账户余额 < 商品价格  
       THEN 取消；abort；
     + 账户余额 -= 商品价格；  
       commit
     + 调用商品运输服务；

1. 请思考：当我们使用Abort撤销一个事务时，是否需要记录相关的日志以确保事务的原子性？如果Abort过程中遇到故障，应该怎么办？

需要一个日志来记录事务已经做了的部分。Abort过程中遇到故障，对undo日志撤销事务已经做的修改无影响。但是Abort结束后一定要写一个Abort日志，然后才能释放加在数据上的锁，才能保证后面的事务运行不出错。

二、在一个同城快递应用的数据库中，有两张表。其中一张为订单表，记录了每一条订单的订单号、送货人信息、收货人信息、货物的信息、发货的起点地址和终点地址、以及送货的状态（包括“新提交订单”、“已接单”、“已取货”、“已送达”四种状态）。另一张为配送表，记录了每个配送员的信息、配送员当前所在的区域、以及配送员正在执行的订单号（当订单号为空时，表示配送员还未接单）。

该应用只允许每个配送员同一时间执行一个订单。该应用对外提供多个功能，包括：

1. 用户下单：用户将订单提交给系统。
2. 配送员接单：配送员浏览自己所在区域的订单，并且选择一个订单接单。
3. 取货：配送员到发货起点取货，并更新订单信息。
4. 完成配送：配送员将货物送达终点，并更新自己和订单的信息。

每次功能调用可能会访问一张或多张表。应用在实际运行时，五种操作会并发进行。此外，运行应用的服务器随时可能遇到故障，我们需要保证：在服务器宕机重启后，用户的订单能够被顺利完成。为此，我们需要使用事务来保证应用的正常运行。例如，用户下单的过程可以按照如下的伪代码执行：

例 功能1：用户下单

Begin Function

同用户进行交互，获得订单的所有信息;

Begin Transaction;

访问订单表：新插入一条元组，标志订单状态为“新提交订单”;

Commit;

End

功能2：配送员接单

Begin Function

获得配送员当前所在的区域;

获得该区域的所有订单;

向配送员展示订单;

配送员选择需要接的单;

配送员提交接单请求;

Begin Transaction;

在订单表中，判断该订单的状态是否仍为“新提交订单”

 IF 仍为“新提交订单” Than {

订单表更新送货人信息，并将送货的状态改为“已接单” ;

配送表更新配送员正在执行的订单号;

Commit；}

Else {abort；Info；}

End

功能3：取货

Begin Function

配送员发送已经取货的请求

Begin Transaction;

在配送表中寻找到配送员正在配送的订单号;

在订单表中将该订单送货的状态改为“已取货” ;

Commit;

End

功能4：完成配送

Begin Function

配送员发送已经配送完成的请求

Begin Transaction;

在配送表中寻找到配送员正在配送的订单号;

在订单表中将该订单送货的状态改为“已送达” ;

在配送表中更新配送员当前所在区域，并将正在执行的订单号更新为空；

Commit;

End