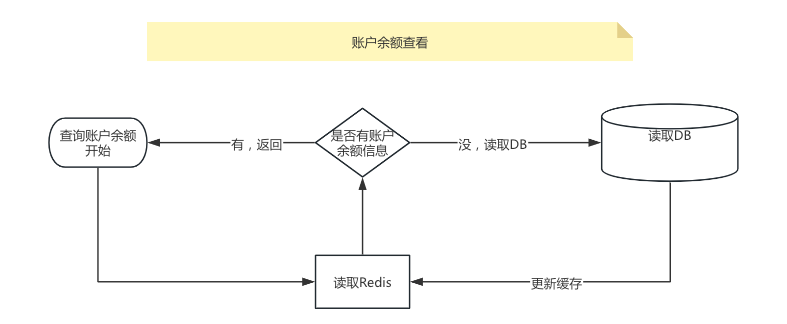
**系统设计**

# 接口设计

## 余额查询接口



## 单个账户余额操作

### 交易的非幂等性

防止客户端重复提交交易，导致多次交易。在每次交易请求时，需要客户端生成唯一的交易ID，同一个交易ID重复提交会提示客户端。

首先在Redis缓存中查询本次交易是否正在执行或者执行完成，如果有，直接返回。

如果没有，可能是缓存已经失效，如果每次去查询数据库来判断交易是否存在，比较耗时，因此在数据库中的交易表transaction\_id的唯一性来决定交易是否存在。

如果可以生成记录，所以没有交易过。如果不能，说明已经交易过。

### 交易的原子性

交易的过程中涉及3张表的写操作，因此需要将三张表的操作放在同一个事务中。

事务的传播：REQUIRED

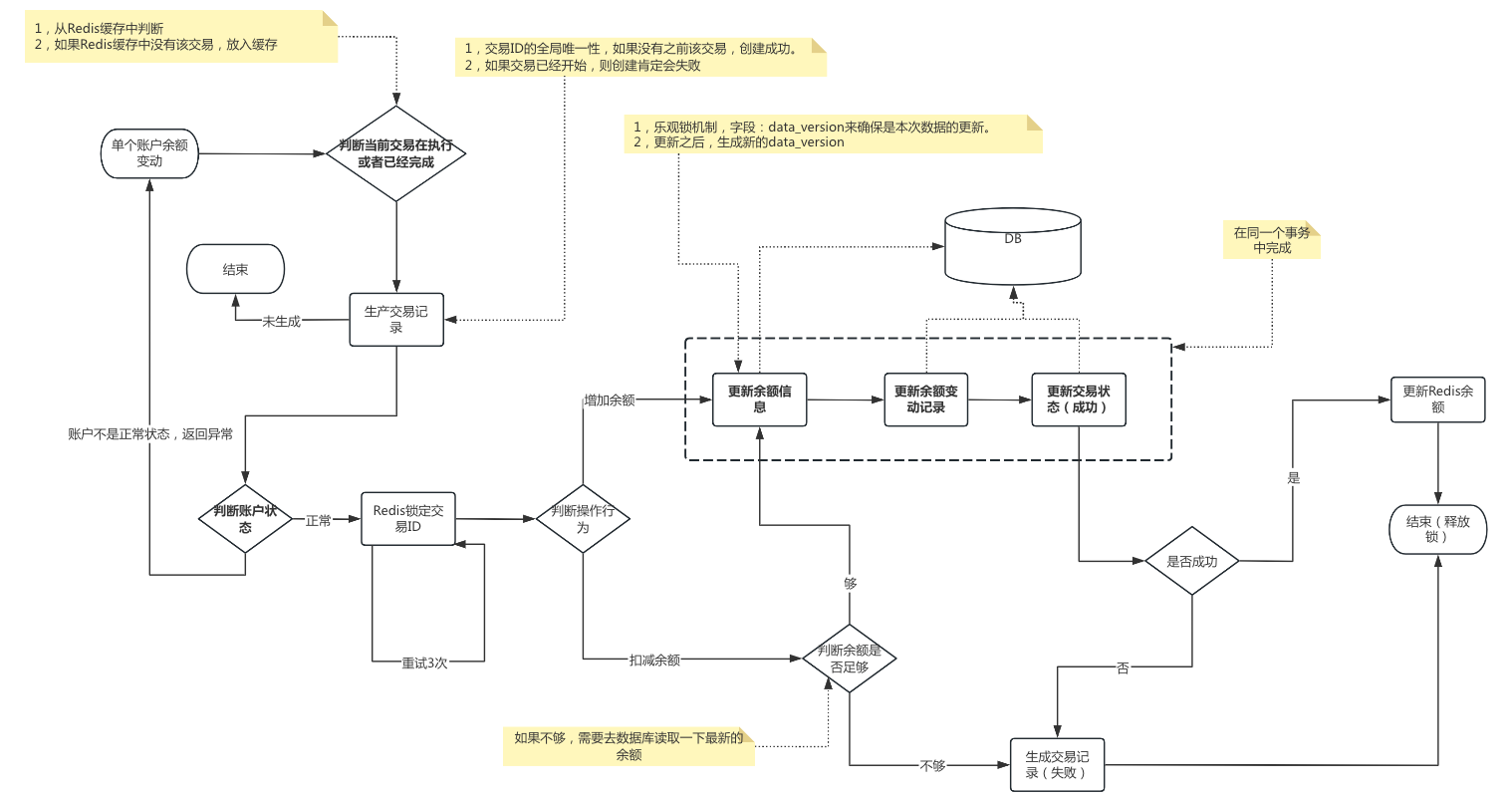
事务的隔离级别： READ\_COMMITTED

有任何异常抛出进行数据回滚。

### 交易的并发性

在同一时刻可能对同一个账户进行并发交易。两个交易获取了用户同样的余额，其中一个交易扣款先完成，另一个交易存款后完成，会导致后面的交易把前面交易给覆盖掉，造成用户余额不对。

1. 悲观锁定账户：每次交易时，锁定该账户【行锁】，等交易完成，释放该锁。该方案可以有效避免交易错乱，但是每次锁定该账户，会造成其他交易（与该账户相关）的等待。
2. 乐观锁定数据版本：给账户表增加一个：data\_version字段，该字段是UUID，每次数据变动之前，读取本次数据版本，在进行数据修改时，需要确保是修改当前data\_version一致的数据，修改赋值新的data\_version。 如果data\_version不一致，修改失败。 该方案有效避免交易的等待。然后通过适当随机的重试机制可以尽量确保并发交易的成功。



### 交易的超时

很可能因为数据库负载的过高，或者网络的一些因素，导致交易迟迟不能完成，从而无法及时的返回给调用方，也难以知道交易的最终状态。

在每次原子交易事务中，需要对整个方法设定事务执行的超时时间，例如超过30秒，认为交易失败，回滚数据。

## 两个账户之间转账

两个账户之间转账涉及表的操作更多一些，但核心的设计思想和单个账户余额操作的一致。

## 交易的状态

每次交易时会生成交易记录，起始的状态是： started，但程序可能：

1. 系统某个时刻崩溃了，但是交易无法进行，状态一直是started。
2. 因为数据库的负载过高，或者网络原因，但是更新数据库迟迟没有反应，状态一直是started.

对于started状态的交易，我们不知道到底是还在执行的过程中，还是其实交易已经挂了。通过以下机制来确保所有的交易在一定时间内都有最终的状态。

1. 在原子操作事务中：BalanceAtomicService ， 在方法的最后，对交易的时长进行判断，如果超过60秒，则认为交易失败，回滚数据。以上保障了只要程序最终执行了，交易一定有状态。 而且可以认为交易太长了，比如超过 3分钟，我们主观上基本可以认定交易失败。
2. 设计一个定时任务，每隔5分钟来检查一下前65分钟到前5分钟内的started状态的交易，将这些超时的60秒的交易重置为：failed
3. 注意：如果因为故障等因素，导致整套系统恢复时间太长，比如一天，就需要人为的去检查那些未完成的交易，根据交易开始时间，判断是否将交易状态重置为：failed

# 数据库设计

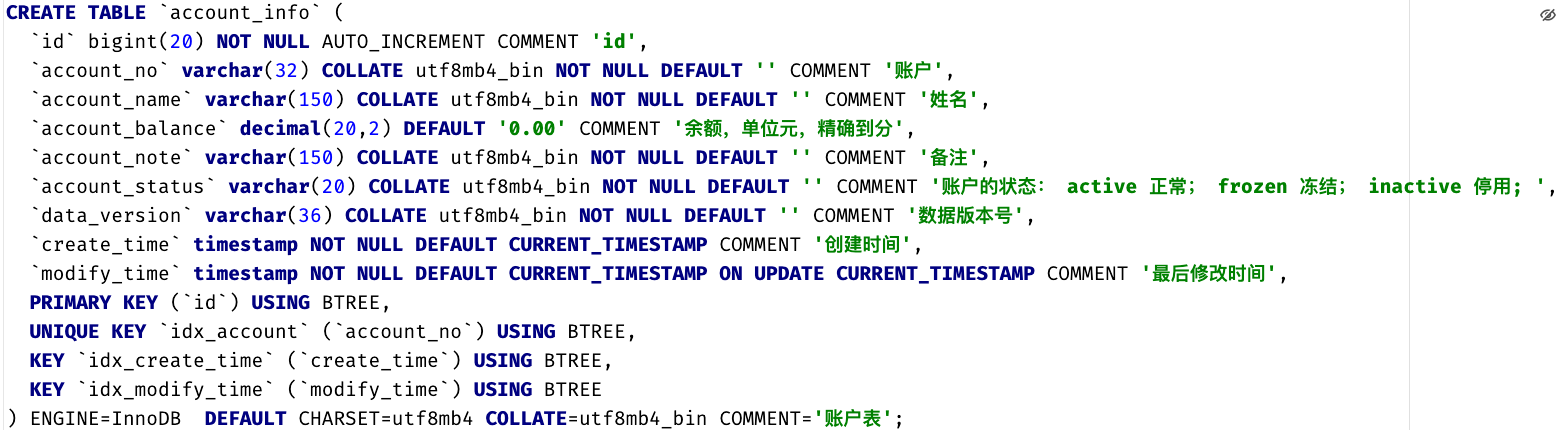
数据库所有表必有三个字段：

id ： Mysql的自增序列，单调递增，作为主键（虚拟）。

create\_time : 创建时间，数据的创建时间，和业务无关。在数据库中生成的时候自动填充当前时间。方便以后排查问题。创建索引。

modify\_time : 最后修改时间，数据的最后修改时间，和业务无关。在数据库中生成的时候自动填充当前时间，而且每次数据修改时，自动更新为当前时间。创建索引。如果需要集成“数仓系统”将依赖该字段。

## 账户表



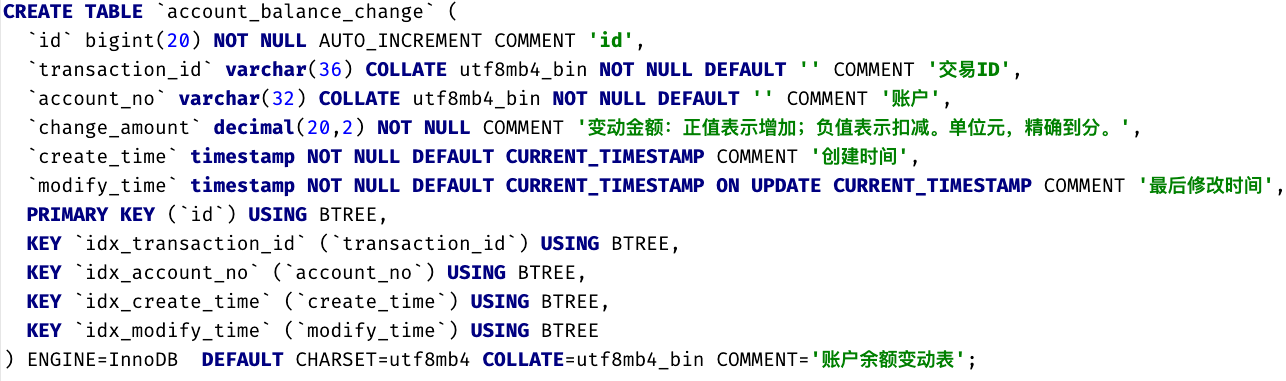
注意：

account\_no： 业务主键，唯一性，构建唯一索引。

account\_status ： 账户的状态

data\_version ： 数据版本,每次数据修改生成新的唯一ID

## 账户余额变动表

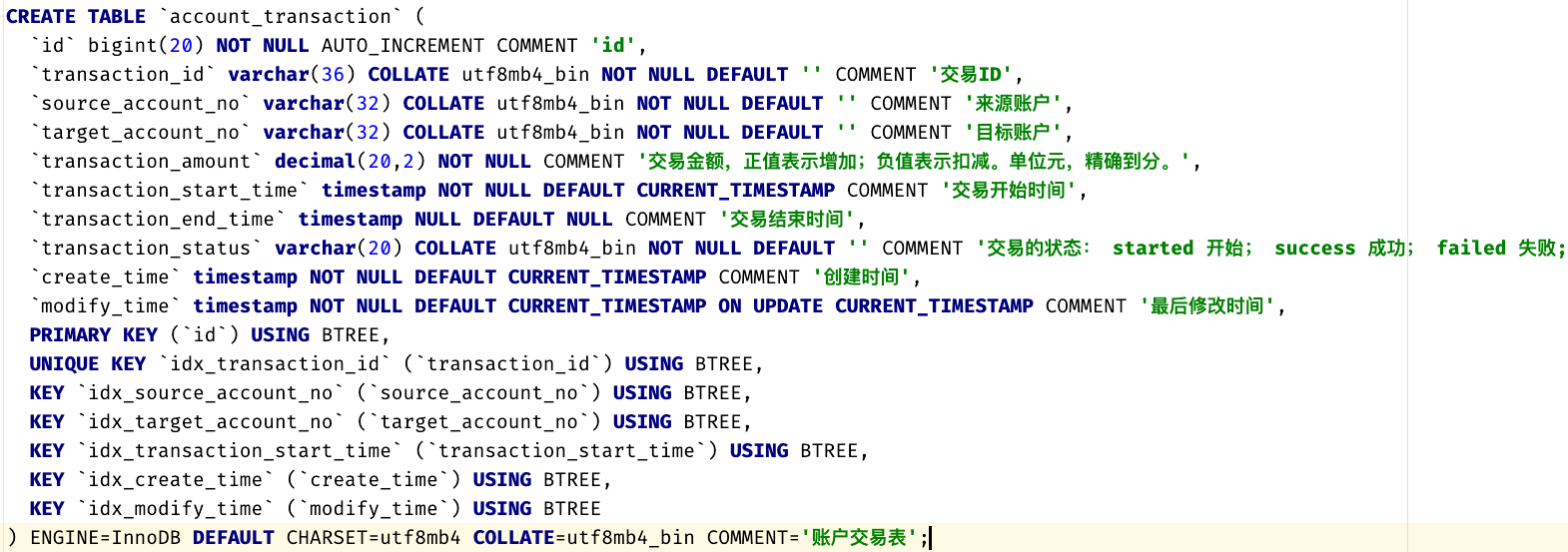


每次账户余额的变动需要生成记录，方便后续进行对账，追溯资金的来源。

transaction\_id : 每次变动，必然是一次交易，有交易ID，构建索引。

account\_no： 账户ID，构建索引。

# 交易表



每次交易生成一条记录，记录交易的信息。

transaction\_id : 交易ID，唯一性。每次交易一条，防止交易重复。唯一索引。

source\_account\_no： 来源账户，如果只是取款/存款，填空。转账才有。构建索引。

target\_account\_no： 对目标账户的操作。构建索引

transaction\_amount： 对目标账户的余额操作金额，正值表示增加；负值表示扣减。单位元，精确到分。

transaction\_start\_time ： 交易开始时间，构建索引，方便后续查询交易。