1.1 数据库事务ACID特性

数据库事务ACID特性:

- 原子性(atomicity)

整个事务中的所有操作要么全部完成,要么全部失败

- 一致性(consistency)

指一个事务可以改变封装状态,事务必须始终保持系统处于一致的状态,不管在任何给定的事件并发事务有多少

- 隔离性(isolation)

它是指两个事务之间的隔离程度

- 持久性(durability)

事务完成以后,该事务对数据库所做的更改便持久保存在数据库之中,并不会被回滚.

1.2 丢失更新问题:

1.第一类丢失更新

时刻	事务一(老公)	事务二(老婆)
T1	查询余额 10000元	
T2		查询余额10000元
T3		网购1000元
T4	吃饭1000元	
T5	提交事务成功, 余额9000元	
T6		不想买了,取消购买,回滚到T2时刻,余额 10000元

这样两个事务并发,一个回滚,一个提交导致不一致,称之为第一类丢失更新,大部分数据库基本都已经消灭了这类更新丢失,一个事务覆盖了另一个事务,可以通过数据库加锁解决

2.第二类丢失更新

时刻	事务一(老公)	事务二(老婆)
T1	查询余额10000元	
T2		查询余额10000元
T3		网购1000元
T4	吃饭1000元	
T5	提交事务成功,查询为10000元,消费1000元后, 余额9000元	
T6		提交事务,根据之前的10000元,扣减 1000元后,余额为9000元

发生在两个事务中,由于在不同的事务中,无法探知其他事务的操作,导致两者都提交后,余额都是9000元,这就是第二类丢失更新.为了克服事务之间协助的一致性,数据库标准规范中定义了事务之间的隔离级别.

1.3 隔离级别

隔离级别可以在不同层度上减少丢失更新,隔离级别定义为4层,分别是脏读(dirty read),读/写提交(read commit),可重复读(repeatable read)和序列化(Serializable)

1.脏读

脏读是最低的隔离级别,其含义是允许一个事务去读取另一个事务中未提交的数据.

时刻	事务一(老公)	事务二(老婆)	备注
T1	查询余额10000元		
T2		查询余额10000元	
T3		网购1000元,余额为9000	
T4	吃饭1000元,余额为8000		读取到事务二,未提交余额为9000元
T5	提交事务成功		余额为8000元
T6		回滚事务,	由于第一类丢失更数据库已经克服,所以余额为错误的8000

2.读/写提交

为了克服脏读,SQL标注提出了第二个隔离级别- 读/写提交,就是说一个事务只能读取另一个事务已经提交的数据

时刻	事务一(老公)	事务二(老婆)	备注
T1	查询余额10000元		
T2		查询余额10000元	
T3		网购1000元,余额为9000	
T4	吃饭1000元,余额为9000		由于事务二未提交,采取读/写提交时不能读出,所以余额为9000元
T5	提交事务		余额为9000元
T6		回滚事务,	由于第一类丢失更数据库已经克服,所以余额为正确的9000

3.不可重读

读/写提交引发的其他问题:

时刻	事务一(老公)	事务二(老婆)	备注
T1	查询余额10000元		
T2		查询余额10000元	
T3		网购1000元,余额为9000	
T4	吃饭2000元,余额为8000		由于事务二未提交,采取读/写提交时不能读出,所以余额为9000元
T5		继续网购8000元,余额1000元	余额为9000元
T6		提交事务,余额为1000元	由于第一类丢失更数据库已经克服,所以余额

			为正确的9000
T7	提交事务发现余额为1000元,不足以买单		由于采取了读/写提交,因此此时的事务一可以知道余额不足

余额是不能重复读取的,而是一个变化的值,这样的场景我们成为不可重复读.消费钱读取10000元,事务二提交后,读取一次1000元,两次数据不一致.

4.幻读

为了克服不可重复读带来的错误,提出了一个可重复读的隔离级别,可重复读震度数据库同一条记录而言的,在很多场景,数据库需要同时对多条记录进行读/写,这个时候就会产生下面的情况.

时刻	事务一(老公)	事务二(老婆)	备注
T1		查询消费记录是10条	初始状态
T2	启用一笔消费		
T3	提交事务		
T4		打印消费记录得到11条	老婆发现打印了11条消费记录,比 查询的10条多一条,她会认为这条多余不存在的 ,这样的场景成为幻读

为了克服幻读,SQL标准提出了序列化的隔离级别,

隔离级别	脏读	不可重复读	幻读
脏读	√	√	√
读/写提交	×	√	√
可重复读	×	×	√
序列化	×	×	×

1.4 选择隔离级别和传播行为

隔离级别出发点在于两点,性能和数据一致性

大部分场景下,企业会选择读/写提交的方式设置事务,有助于提高并发,又压制了脏读.

1.5传播行为

传播行为是指方法之间的调用事务策略问题,大部分情况下,我们希望事务能够同时成功,或者同时失败,一个方法调度另一个方法时,可以对事务特性进行传播配置,称为传播行为

spring的7种传播行为:

传播行为	含义	备注
REQUIRED	当方法调用时,如果不存在当前事务那么就创建事务,如果之前方法已经存在事务,那么久沿用之前的事务	这是spring默认的传播行为

SUPPORTS	当方法调用时,如果不存在当前事务,那么不启用事务,;如果存在当前事务,那么就沿用当前事务	
MANDATORY	方法必须在事务内运行	如果不存在当前事务,则抛出异常
REQUIRES_NEW	无论是否存在当前事务,方法都会在新事物中运行	也就是事务管理器会打开新的事务运行该方法
NOT_SUPPORTED	不支持事务,如果不存在当前事务也不会创建事务;如果存在当前事务,则挂起它,直至该方法结束后才恢复当前事务	适用于那些不需要事务的SQL
NEVER	不支持事务,只有在没有事务的环境中才能运行它	如果方法存在当前事务,则抛出异常
NESTED	嵌套事务,也就是调用方法如果抛出异常只回滚自己内部的SQL,而不回滚主方法的事务	它的实现存在两种情况,如果当前数据库支持保存点(savepoint),那么它就会在当前事务上使用保存点技术;如果发生异常,则将方法执行的SQL回滚到保存点,而不是全部回滚,否则就等同于REQUIRES_NEW创建新的事务运行方法代码