事务

数据库事务是什么? 为啥要有事务这么个东西?

最经典的转账场景, A给B转账100块钱. 你就得写两个sql, A账户减少100块钱, B账户增加100块钱. 那给B账户里加钱失败了怎么办? A不就平白无故少了100块钱.

在这个转账场景中, 这两个sql应该是一个整体, 要么都成功要么都失败, 如果一个成功了, 一个失败了那数据就有问题了.

所以就需要有数据库事务. 所谓事务就是一系列SQL语句构成一个逻辑上的整体, 要么全部成功, 要么全部失败.

回到刚才的例子, A账户减少100块钱执行成功, B账户增加100块钱执行失败, 那么回滚对A账户的操作. 这样, 两个操作就变成了一个整体.

事务四大特性

原子性

- 事务是不可分割的最小单元, 这就是刚才说的要么都成功, 要么都失败, 不会在某个没执行成功的中间态结束.
- 原子性通过undolog回滚日志来实现的, 比如n个连续操作失败了一个, 通过undolog让前面的操作全部回滚.

持久性

- 持久性是指一个事务一旦被提交， 它对数据库中数据的改变就是永久性的, 无法撤销. 即便数据库崩溃了, 也能保证修改不丢失.

- 持久性是redolog来实现的, 每一次事务提交后, 会把修改写入redolog, 即便mysql数据没有持久化到磁盘中, 也能读redolog恢复数据. 所以redolog保证了事务的持久性. 这就是为什么说redolog让mysql有了崩溃恢复用的能力.
- 那一定有人在想, redolog自己本身也是持久化到磁盘的, 它也是要刷盘的, 要是mysql崩了, redolog也刷盘失败了, 那数据不就丢了. 这个问题在日志详细聊.

隔离性

- 并发访问数据库时, 一个用户的事务不会被其他事务的操作所干扰. 多个并发事务之间相互隔离, 每个事务不受并发影响, 独立执行.
- 如果事务A把商品库存从0修改为1, 事务B一读, 发现是1准备扣减库存. 但是事务A回滚了, 又把商品库存回滚到0了. 那事务B再做扣减库存的操作不就会出问题吗. 这就是脏读, 一个事物读取到了另一个事务没有提交的数据.
- 这也是事务隔离性要解决的问题, 要让各个事务独立执行, 不相互影响.
- 隔离性会, mvcc+锁 配合undolog来实现. 这个后面会详细讲.

一致性

- 执行事务前后，数据总量要保持一致.
- 例如转账业务中，无论事务是否成功，转账人和收款人的总额应该是不变的.
- 一致性怎么实现呢? 其实只要保证了原子性, 持久性, 隔离性三个特性, 一致性就自然实现了.

并发事务带来的问题

事务的并发执行会带来很多问题, 刚才将隔离性的时候提到了脏读, 除了脏读, 还有不可重复度, 幻读.

- 脏读
  - 一个事务读取到其他事务未提交的数据.
- 不可重复度
  - 当事务A第一次读值为1, 然后事务B修改值为2, 事务B提交. 然后事务A第二次读发现值为2了.
  - 在同一个事务中，前后两次读取的数据不一致, 这是不可重复读.
- 幻读
  - 当事务A第一次做范围查询, 查询id大于100的数据有10条, 此时事务B又插入了20数据, 事务A再次范围查询发现id大于100的有30条了. 这就是幻读.
  - 幻读和不可重复读有些类似，不可重复度强调多次读取数据值不一样, 但是幻读强调的是数据条数的增减, 而不是数据值的更新.

事务的隔离级别

那怎么解决脏读, 不可重复度和幻读的问题呢?

- 隔离级别. 数据库提供了四种事务的隔离级别, 读未提交、读已提交、可重复读、串行化.

读未提交

- 最低的隔离级别, 每个事务都可以看到其他事务未提交的数据.

- 读未提交不能解决脏读，可重复读，幻读的问题，所以实际不会用.

读已提交

- 每个事务只能读到其他事务已经提交的数据.
- 这就可以解决脏读的问题, 因为其他事务没提交的你读不到. 但是无法解决不可重复度和幻读的问题.

可重复读

- mysql默认的隔离级别.
- 在可重复读这种隔离级别下, 当第一次读的时候, 会生成一份数据快照. 生成快照后, 其他事务的修改对当前事务来说是不可见的.
- 因为你每次都读这个快照, 就可以保证在同一个事务内两次读到的数据是一样的. 所以就解决不可重复读的问题, 但是无法解决幻读问题.

串行化

- 最高的隔离级别
- 串行化要求事务不能并发执行, 必须一个接一个顺序执行. 你都无法并发执行了, 那些并发问题自然而然就不存在了.
- 所以串行化可以解决所有并发问题, 但是性能很差, 并发量高的情况下可能会导致大量的超时和锁竞争问题, 通常不会用这个隔离级别.

所以事务隔离级别等级越高，就越能保证数据的一致性和完整性，但是执行效率也越低. 事务的隔离级别越低, 执行效率就越高, 但是数据的一致性就越差.

MySQL 的隔离级别怎么实现的

那现在已经有四种隔离级别能解决问题了, 那应该怎么实现这几种隔离级别呢?

- 读未提交怎么实现?
  - 不用实现, 你根本不用管, 多个并发事务同时执行天然就是读未提交. 当然这里说的是读的时候你不用管, 写的时候还是要加锁的, 不能同时写一个数据呀.
- 串行化怎么实现的?
  - 这就很简单了, 直接加锁就行了. 只有能拿到锁的事务才能执行, 这样事务就串行化执行了.

那读已提交和可重复读怎么实现, 这就涉及到了MySQL中一个相当复杂但是又很常考的东西, MVCC.

由于这玩意儿确实太复杂, 所以MVCC详细的讲解放到下一期, 这里简单讲一下MVCC.

- MVCC多版本并发控制, 就是维护数据的多个版本, 怎么维护的, 通过**undolog版本链**.
- MySQL表里的一行数据除了你能看到的数据外, 还有几个隐藏字段, 其中一个叫事务id, 当某事务修改数据时, 就把该事务的id记录在这个隐藏字段中.
- 另一个隐藏字段叫回滚指针, 这个回滚指针就指向undolog版本链, 也就是说通过回滚指针就能在undolog版本链中查到这个数据的各个版本.
- 比如说, 某数据第一个版本是1, 有事务将其修改为2, 就产生了第二个版本, 第二个版本值为2, 又有事务过来将其修改为3, 就产生了第三个版本, 第三个版本值为3.

- 那么问题来了, 现在有个事务需要读这个数据, 那么它读到的是哪个版本, 是1, 还是2, 还是3.
- 它读到的是哪个版本, 取决于readview. 执行select时会生成一个快照, 叫readview. readview中记录了一堆事务id, 主要是用来判断数据版本对当前事务的可见性.
- 简单来说只要对比当前事务生成的readview和数据隐藏字段的事务id, 就能知道那个版本的数据对当前事务是可见的. 如果不可见, 那就跟着回滚指针沿着版本链往下找, 找到当前事务可见的数据版本, 然后返回即可.

所以一句话总结mvcc, 每次修改数据就记录修改的事务id到隐藏字段, 然后生成一个版本记录到undolog版本链中, 然后回滚指针指向版本链, 通过对比readview和事务id就能知道某个版本的数据对当前事务是否可见. 一定要记住, readview就是MVCC中用来判断数据版本的可见性的, 这个非常重要. 一定有人问, readview咋判断数据可见性的, 这个比较复杂, 下一期详细讲.

接下来看看可重复读和读已提交怎么实现的

- 可重复读怎么实现的?
    - 可重复读, 每次都会读到和第一次相同的数据, 即便后面有新事务修改了数据, 还是会读到和第一次相同的数据, 所以解决了不可重复读的问题.
    - 那么问题来了, 怎么让他每次都读到相同的数据呢?
    - 可重复读会在第一次select的时候生成一个readview, 刚说了, readview是用来判断数据版本对当前事务的可见性的, 可重复读每次都会复用第一次生成的readview.
    - 也就是说, 当生成这个readview的时候, 哪些版本的数据对当前事务可见就已经固定下来了, 后续每次通过同一个readview判断数据可见性, 所以每次都读的是同一个数据版本.

    - 所以可重复读就是通过MVCC+复用第一次生成的readview来实现的. 除此之外, 可重复读为了避免幻读问题, 还会加锁. 这个下期再详细聊.
- 读已提交怎么实现的?
    - 读已提交就是说, 第一次读一些数据, 后面只要有事务修改了数据, 然后提交了, 那再读就能读到最新的已经提交的数据.
    - 读已提交的本质就是每次select就生成一个新的readview, 每次读都生成新的readview, 就能每次读到新提交的数据版本.