你真的理解mysql的事务隔离吗？

原创 fliyu 邂逅Go语言 2023年03月01日 23:38 广东

事务

提到事务，你很容易想到数据库的四个特性，即ACID，也被称为事务的原子性、一致性、隔离性和持久性。这些特性确保了事务在数据库中正确执行，并确保了数据的完整性和一致性。

1. **原子性(Atomicity)**：指一个事务是一个不可分割的单位，事务中的所有操作要么全部完成，要么全部不完成，不能只完成其中一部分。如果事务执行过程中出现错误，那么事务将被回滚到最初的状态，撤销所有更改，保证数据一致性。
2. **一致性(Consistency)**：指事务的执行结果必须是数据库从一个一致性状态变到另一个一致性状态。在事务执行过程中，数据的完整性约束不能被破坏。
3. **隔离性(Isolation)**: 指在并发环境中，当多个事务同时执行时，每个事务都应该是相互隔离的，互不干扰，不会出现数据冲突等。
4. **持久性(Durability)**: 指当前事务完成后，对于数据的修改是永久性的，不会因为系统故障等原因造成数据丢失。数据库应该能够保证事务所做的修改即使在系统故障时也能够永久保存下来。

这篇文章我们主要来讲讲其中的隔离性(Isolation)。

隔离级别

SQL标准的事务隔离级别有四种，从低到高分别是：**读未提交(read uncommitted)**、**读提交(read committed)**、**可重复读(repeatable read)** 和**串行化(serializable)**，事务的隔离级别越高，隔离的程度就越严实，效率自然而然也就越低。

读未提交：顾名思义，就是一个事务可以读取另一个未提交事务的数据，一个事务还没提交时，它所做的变更就能被其他事务看到。

读提交：顾名思义，只能读取到已经提交的数据，一个事务提交之后，它做的变更才会被其他事务看到。

可重复读：指一个事务在执行过程中看到的数据，总是跟这个事务最开始看到的数据一致。它是Mysql的默认隔离级别。

串行化：顾名思义，事务串行化的按顺序执行，后访问的事务必须等前一个事务执行完成，才能继续执行，这是数据库最高的隔离级别，同时也是性能最差的。

在不同的隔离级别下，事务所能够访问的范围是不一样的，自然就有了取舍。同时，每种隔离级别又带来了不一样的问题，如下表所示：

隔离级别	脏读	不可重复读	幻读
读未提交	可能	可能	可能
读提交	不可能	可能	可能
可重复读	不可能	不可能	可能
串行化	不可能	不可能	不可能

什么是脏读、不可重复读、幻读？

脏读：指一个事务读到另外一个事务已修改但未提交的数据时，这种情况被称为脏读。

不可重复读：指在一个事务内，多次读取同一数据，前后读取到的数据是不一样的，这种情况被称为不可重复读。

幻读：指一个事务先后读取一个范围的数据，但两次读取到的记录数不同（两次执行同一条select语句），例如增加了一行记录或者减少了一行记录，这种情况被称为幻读。

理解“读提交”和“可重复读”

事务A	事务B
begin;	begin;
select v from t; // 查询得到数值1	
	select v from t; // 查询得到数值1
	update t set v = v+1; // 将1改为2
select v from t; // 查询得到数值V1	
	commit;
select v from t; // 查询得到数值V2	
commit;	
select v from t; // 查询得到数值V3	

如果隔离级别是读提交，则V1的值是1，因为当前事务B所做的更新还未提交，只有事务B的更新在提交后，才能被其他事务看见，因此V2、V3的值都为2。

如果隔离级别是可重复读，在事务A开始执行时，就产生了快照，所以在整个事务执行过程中看到的数据前后必须是一致的，因此V1、V2的值都是1，V3是2。

总结读提交和可重复读的逻辑区别：

在可重复读隔离级别下，只需要在事务开始的时候创建一致性视图，整个过程保持一致，之后事务里的其他查询都共用这个一致性视图。

在读提交隔离级别下，事务里的每一行语句执行前都会重新计算出一个新的视图。

事务隔离的实现：MVCC多版本并发控制

MVCC是 mysql 等数据库管理系统实现事务隔离性的一种方式。它基于在事务中读取和写入数据时使用版本号来实现数据的多版本控制，在InnoDB中，会在每行数据后添加两个额外的隐藏的值来实现MVCC，这两个值一个记录这行数据何时被创建，另外一个记录这行数据何时过期（或者被删除）。在MVCC中，每个事务可以看到不同的数据库快照

版本，这使得每个事务看到的数据都是一致的，即使其他事务正在同时修改数据。

MVCC使用了两个重要的技术：读取视图和版本链。读取视图是每个事务的“快照”，它定义了事务所能看到的数据版本。版本链则是用于跟踪每个数据行的历史版本，当事务修改数据时，新版本会被创建并与之前的版本链接在一起。

MVCC这种读取历史数据的方式称为快照读(snapshot read)，而读取数据库当前版本数据的方式，叫当前读(current read)。

快照读：mysql 中默认使用的select语句就是快照读，这样能够减少加锁带来的开销。

当前读：能够对数据进行修改的语句都是采用当前读的模式（例如：insert、update、delete），在执行这几个操作时，都会读取到最新的记录，即别的事务已经提交的数据。同样的，select语句如果加锁，也是当前读：

```
select * from table where id = 1 lock in share mode;
select * from table where id = 1 for update;
```

事务的隔离性

在mysql中，默认隔离级别是可重复读，事务A启动的时候创建一个视图，之后事务A在执行期间，即使有其他事务修改了数据，事务A看到的数据仍然是跟刚开始启动的时候一样，也就是说在可重复读隔离级别下，数据不会受到其他事务的影响。

下面我以可重复读隔离级别看看事务的隔离性：

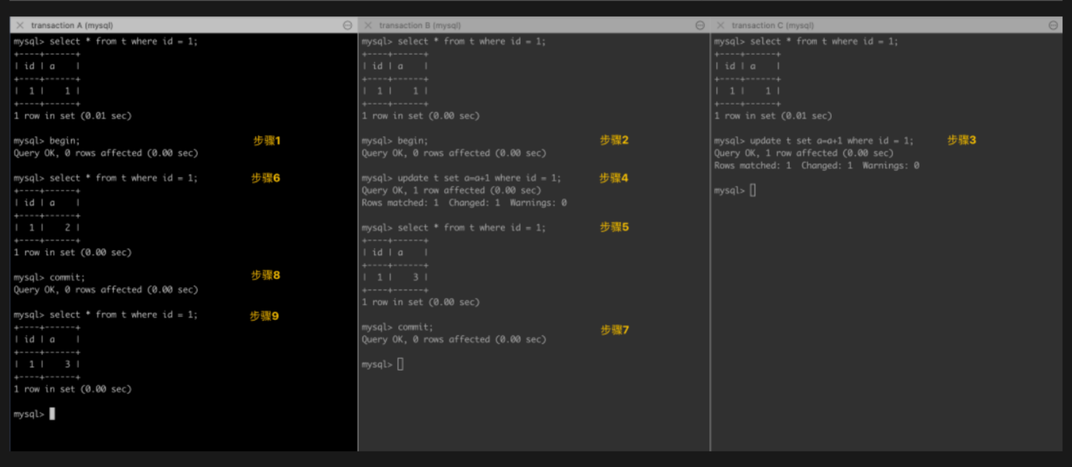
start transaction或begin是在第一条select执行完后，才得到事务的一致性快照。

start transaction with consistent snapshot则是立即得到事务的一致性快照。

```
mysql> CREATE TABLE `t` (
  `id` int(11) NOT NULL,
  `a` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;
mysql> insert into t(id, a) values(1,1),(2,2);
```

在start transaction 下：

事务A	事务B	事务C
begin;		
	begin;	update t set a=a+1 where id=1;
	update t set a=a+1 where id=1;	
	select a from t where id = 1;	
select a from t where id = 1;		
commit;		
	commit;	
select a from t where id = 1;		



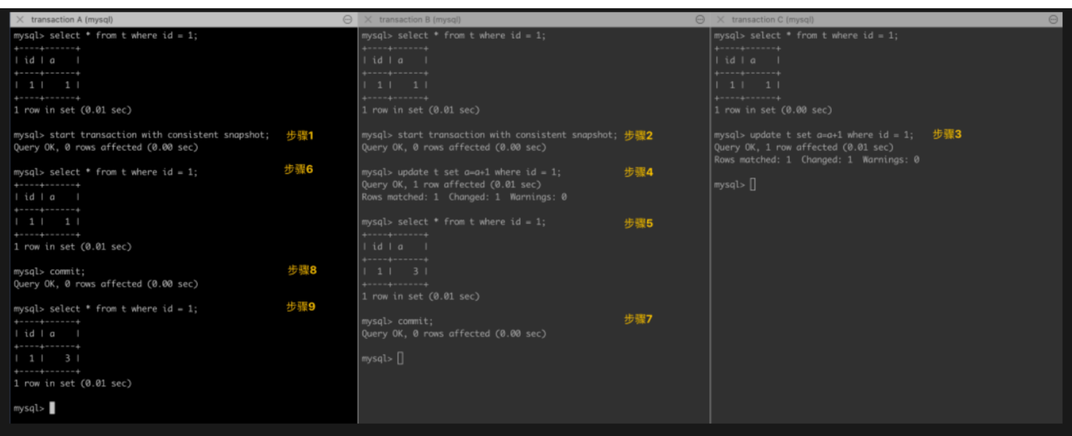
来分析下最后的结果：

事务A第一次查询出来的值是2(步骤6)，因为begin是在第一条select语句执行完后才得到事务的一致性快照，在select语句执行前，事务C的update操作(步骤3)就已经完成了，而事务B的update(步骤4)操作还没提交，因此事务A在还没获取快照一致性前能够察觉到已经提交的事务。

事务B第一次查询出来的值是3(步骤5)，为什么不是2呢，因为事务C的update操作(步骤3)已经完成了，而update语句使用的是当前读，如果使用快照读，会把事务C的操作给覆盖了。

在start transaction 下：

事务A	事务B	事务C
start transaction with consistent snapshot;		
	start transaction with consistent snapshot;	update t set a=a+1 where id=1;
	update t set a=a+1 where id=1;	
	select a from t where id = 1;	
select a from t where id = 1;		
commit;		
	commit;	
select a from t where id = 1;		



事务A第一次查询出来的值是1(步骤6)，因为start transaction with consistent snapshot;(步骤1)是在命令一完成就获取了事务的一致性快照，而事务B、事务C的操作都比步骤1慢，因此步骤6的值就是1，一旦commit提交后，当前事务就结束了，也就能获取到最新的值了。

事务B的操作就跟上述的一样了。

第一种启动方式:一致性视图是在执行第一个快照读语句时创建的；

第二种启动方式:一致性视图是在执行 start transaction with consistent snapshot 时创建的。

可重复读的核心就是一致性读，事务在更新数据的时候，只能用当前读。如果当前的记录的行锁被其他事务占用的话，就需要进入锁等待。

事务隔离的原理

如果事务启动的时候，就把数据库克隆一份生成快照，那得有多慢。

前面我们讲过，数据库是使用MVCC多版本并发控制机制实现事务的隔离性，在事务中读取和写入数据时使用版本号来实现数据的多版本控制，在MVCC中，每个事务可以看到不同的数据库快照版本，这使得每个事务看到的数据都是一致的。

InnoDB 里面每个事务有一个唯一的事务 ID，叫做 transaction id。它是在事务开始的时候向 InnoDB 的事务系统申请的，是按申请顺序严格递增的。

而每行数据也都是有多个版本的。每次事务更新数据的时候，都会生成一个新的数据版本，并且把 transaction id 赋值给这个数据版本的事务 ID，记为 row trx_id。同时，旧的数据版本要保留，并且在新的数据版本中，能够有信息可以直接拿到它。

在上图中，事务启动的顺序是自增的，语句更新时都会生成undo log日志，当需要获取前一个版本时，就是根据undo log 日志倒算出来的。

总结：

本篇文章主要讲了mysql数据库事务的隔离，涉及到的概念很多，还是要多加思考琢磨，在日常使用中也是会用到的，更重要的是面试大概率会问题。

一百分以内，一分耕耘一分收获；

一百分以外，一分耕耘十分收获。

mysql 2 # 数据库 1

mysql · 目录 ≡

下一篇 · Go MySQL 互相喜欢怎么实现？ >