23 细聊分布式事务

更新时间：2019-10-01 20:54:22

> 耐心是一切聪明才智的基础。
>
> ——柏拉图

比如你在网上买了一本书，可以简化为在订单库增加订单，在库存库减掉这本书的 1 个库存。这里订单库和库存库是在不同的机器上，如果这两步放在两个事务里，增加订单这一步成功了，但是减库存这里失败了。那岂不是就乱了。

这里就要引出分布式事务了。什么是分布式事务？

1 认识分布式事务

分布式事务是指一个大的事务由很多小操作组成，小操作分布在不同的服务器上或者不同的应用程序上。分布式事务需要保证这些小操作要么全部成功，要么全部失败。MySQL 从 5.0.3 开始支持分布式事务。

分布式事务使用两阶段提交协议：

- 第一阶段：所有分支事务都开始准备，告诉事务管理器自己已经准备好了；
- 第二阶段：确定是 rollback 还是 commit，如果有一个节点不能提交，则所有节点都要回滚。

与本地事务不同点在于：分布式事务需要多一次 prepare 操作，等收到所有节点的确定信息后，再进行 commit 或者 rollback。

上面买书的例子，就可以放到一个分布式事务里，保证增加订单和减库存操作有原子性，要么全部成功，要么全部失败。

MySQL 中分布式事务按实现方式可以分为两种：MySQL 自带的分布式事务和结合中间件实现分布式事务。下面来详细介绍一下这两种分布式事务。

2 MySQL 自带的分布式事务

MySQL 有自带的分布式事务实现方法，具体语法如下：

启动分支事务：

```
xa start 'a','a_1';
```

> ‘a’,‘a_1’ 表示 xid，
>
> a 表示 gtrid，为分布式事务标识符，相同的分布式事务使用相同的 gtrid。
>
> a_1 表示 bqual，为分支限定符，分布式事务中的每一个分支事务的 bqual 必须不同。

结束分支事务：

```
xa end 'a','a_1';
```

进入准备状态：

```
xa prepare 'a','a_1';
```

提交分支事务：

```
xa commit 'a','a_1';
```

回滚分支事务：

```
xa rollback 'a','a_1';
```

返回当前数据库中处于 prepare 状态的分支事务的详细信息：

```
xa recover;
```

我们来看一个具体例子：

session1	session2
use muke1;	use muke2;
create table t23_1(id int);	create table t23_2(id int);
xa start ‘test’,‘muke1’;	xa start ‘test’,‘muke2’;
insert into t23_1 select 1;	insert into t23_2 select 1;
xa end ‘test’,‘muke1’;	xa end ‘test’,‘muke2’;
xa prepare ‘test’,‘muke1’;	xa prepare ‘test’,‘muke2’;
xa recover \G	xa recover \G
xa commit ‘test’,‘muke1’;	xa commit ‘test’,‘muke2’;

上面的例子就演示了一个分布式事务，事务在 muke1 库中的 t23_1 表中插入一条记录，同时在 muke2 库中的 t23_2 表中插入一条记录，两个操作作为同一个事务提交。在进入准备状态之前，如果 session2 中某一步没执行成功而回滚了，则 session1 和 session2 整个分布式事务的操作都会回滚。

但是 MySQL 5.7 之前的版本，自带的分布式事务存在以下问题：

比如某个分支事务到达 prepare 状态时，此时数据库断电，重启后，可以继续对分支事务进行提交或者回滚，但是提交的事务不会写 binlog，如果有从库，会导致主从数据不一致的情况。

如果分支事务的客户端连接异常中止，那么数据库会自动回滚当前分支未完成的事务，如果此时分支事务已经到 prepare 状态，那么这个分布式事务的其他分支可能已经成功提交，如果这个分支回滚，可能导致分布式事务的不完整，丢失部分分支事务的内容。

还有一种情况，如果分支事务在执行到 prepare 状态时，数据库出现故障，并且无法启动，需要使用全备和 binlog 来恢复数据，那么这些在 prepare 状态的分支事务因为没有记录到 binlog，所以也不能通过binlog 进行恢复，在数据库恢复后，将丢失这部分数据。

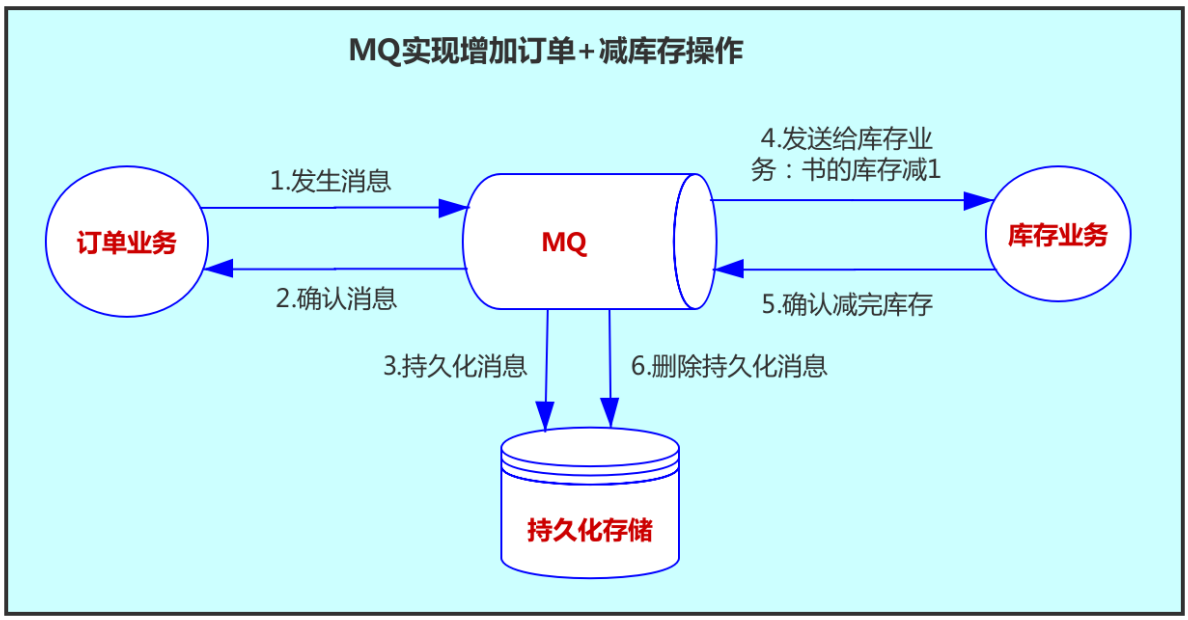
所以，MySQL 5.7 之前的版本自带的分布式事务还存在比较严重的缺陷，在有些场景下，会导致数据丢失。如果业务对数据完整性要求不改，可以考虑使用，如果对数据完整性要求比较高，需要考虑先升级到 5.7 版本。

3 结合中间件实现分布式

上面说了 MySQL 自带的分布式事务，这里再介绍一下借助中间件实现分布式的情况。

具体实现方式可以拿上面网上购书的例子来说：

订单业务程序处理完增加订单的操作后，将减库存操作发送到消息队列中间件中（比如：Rocketmq），订单业务程序完成提交。然后库存业务程序检查到消息队列有减对应商品库存的信息，就开始执行减库存操作。库存业务执行完减库存操作，再发送一条消息给消息队列中间件：内容是已经减掉库存。具体步骤如下：



当然，为了确定最终已经完成减库存操作，还可以加一步对数据库中该商品库存的判断。

4 总结

本节讲解了分布式事务，所谓分布式事务，是指一个大的事务由很多小操作组成，小操作分布在不同的服务器上或者不同的应用程序上。分布式事务需要保证这些小操作要么全部成功，要么全部失败。

本节讲解了两种分布式方式：

- MySQL 自带的分布式事务
- 结合中间件实现分布式

当然，目前主流的分布式实现还是结合中间件实现分布式处理的，本节也举例说明了使用 MQ 实现分布式的例子。

5 问题

各位朋友是通过什么方式实现分布式的呢？

6 参考资料

《高性能 MySQL》第 3 版 7.11 分布式（XA）事务

《深入浅出 MySQL》第 2 版 14.3 分布式事务的使用

《MySQL 技术内幕：InnoDB存储引擎》第 2 版 7.7 分布式事务

}

22 养成好的事务习惯

24 如何预防SQL注入?