MySQL悲观锁、乐观锁总结

概念区分

1. 乐观锁（Optimistic Locking）： 顾名思义，对加锁持有一种乐观的态度，即先进行业务操作，不到最后一步不进行加锁，"乐观"的认为加锁一定会成功的，在最后一步更新数据的时候再进行加锁。
2. 悲观锁（Pessimistic Lock）： 正如其名字一样，悲观锁对数据加锁持有一种悲观的态度。因此，在整个数据处理过程中，将数据处于锁定状态。悲观锁的实现，往往依靠数据库提供的锁机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了加锁机制，也无法保证外部系统不会修改数据）。

悲观锁

1.数据表中的实现
在MySQL中使用悲观锁，必须关闭MySQL的自动提交，**set autocommit=0**，MySQL默认使用自动提交autocommit模式，也即你执行一个更新操作，MySQL会自动将结果提交。

举例：

```
//step1: 查出商品剩余量
select quantity from items where id=100;

//step2: 如果剩余量大于0，则根据商品信息生成订单
insert into orders(id,item_id) values(null,100);

//step3: 修改商品的库存
update Items set quantity=quantity-1 where id=100;
```

这样子的写法，在小作坊真的很正常，No Problems，但是在高并发环境下可能出现问题。
其实在step1或者step2环节，已经有人下单并且减完库存了，这个时候仍然去执行step3，就造成了超卖。

但是使用悲观锁，就可以解决这个问题，在上面的场景中，商品信息从查询出来到修改，中间有一个生成订单的过程，使用悲观锁的原理就是，当我们在查询出items信息后就把当前的数据锁定，直到我们修改完毕后再解锁。那么在这个过程中，因为数据被锁定了，就不会出现有第三者来对其进行修改了。而这样做的前提是需要将要执行的SQL语句放在同一个事物中，否则达不到锁定数据行的目的。

使用悲观锁：

```
//step1: 查出商品状态
select quantity from items where id=100 for update;

//step2: 根据商品信息生成订单
insert into orders(id,item_id) values(null,100);

//step3: 修改商品的库存
update Items set quantity=quantity-2 where id=100;
```

select...for update是MySQL提供的实现悲观锁的方式。此时在items表中，id为100的那条数据就被我们锁定了，其它的要执行select quantity from items where id=100 for update的事务必须等本次事务提交之后才能执行。这样我们可以保证当前的数据不会被其它事务修改。

需要注意的是，当我执行select quantity from items where id=100 for update后。如果我是在第二个事务中执行select quantity from items where id=100（不带for update）仍能正常查询出数据，不会受第一个事务的影响。另外，MySQL还有个问题是select...for update语句执行中所有扫描过的行都会被锁上，因此在MySQL中用悲观锁务必须确定走了索引，而不是全表扫描，否则将会将整个数据表锁住。

悲观锁并不是适用于任何场景，它也存在一些不足，因为悲观锁大多数情况下依靠数据库的锁机制实现，以保证操作最大程度的独占性。如果加锁的时间过长，其他用户长时间无法访问，影响了程序的并发访问性，同时这样对数据库性能开销影响也很大，特别是对长事务而言，这样的开销往往无法承受，这时就需要乐观锁。

乐观锁

一般是在数据表中加上一个数据版本号version字段，表示数据被修改的次数，当数据被修改时，version值会加一。当线程A要更新数据值时，在读取数据的同时也会读取version值，在提交更新时，若刚才读取到的version值为当前数据库中的version值相等时才更新，否则重试更新操作，直到更新成功。

举例：

```
//step1: 查询出商品信息
select (quantity,version) from items where id=100;

//step2: 根据商品信息生成订单
insert into orders(id,item_id) values(null,100);

//step3: 修改商品的库存
update items set quantity=quantity-1,version=version+1 where id=100 and version=#
{version};
```

使用场景

- 悲观锁：比较适合写入操作比较频繁的场景，如果出现大量的读取操作，每次读取的时候都会进行加锁，这样会增加大量的锁的开销，降低了系统的吞吐量。
- 乐观锁：比较适合读取操作比较频繁的场景，如果出现大量的写入操作，数据发生冲突的可能性就会增大，为了保证数据的一致性，应用层需要不断的重新获取数据，这样会增加大量的查询操作，降低了系统的吞吐量。

特点

- 悲观锁：悲观锁的特点是先获取锁，再进行业务操作，即“悲观”的认为获取锁是非常有可能失败的，因此要先确保获取锁成功再进行业务操作。通常所说的“一锁二查三更新”即指的是使用悲观锁。
- 乐观锁：乐观锁的特点先进行业务操作，不到万不得已不去拿锁。即“乐观”的认为拿锁多半是会成功的，因此在进行完业务操作需要实际更新数据的最后一步再去拿一下锁就好。

总结

读取频繁使用乐观锁，写入频繁使用悲观锁。乐观锁不能解决脏读的问题