悲观锁和乐观锁详解

##### 背景

考虑下面两个并发带来的问题：

1、丢失更新：一个事务的更新结果覆盖了其它事务的更新结果，即所谓的更新丢失。

2、脏读：当一个事务读取其它完成一半事务的记录时，就会发生脏读取。

例如：

两个用户同时修改商品库存表，A、B同时进入，看到的库存都是100，A购买一件把库存修改为99(100-1)。此时B购买两件把库存修改为98(100-2)，因为A、B同时读到的库存都是100，B并不能看到A做的库存更新，所以造成B脏读，造成A丢失更新。

因此为了解决这些并发带来的问题。 我们需要引入并发控制机制--锁。

##### 锁分类

**悲观锁**

**悲观锁就是用户修改数据时看起来很悲观，保守态度，担心别的用户会同时修改这条数据，所以每次修改时会提前把这条数据锁定起来，只有自己可修改（但别的用户可以读），等自己修改完了再释放锁。**

**乐观锁**

**乐观锁就是用户修改数据时心态很乐观，不管别人修改不修改数据，我都不上锁，我修改的时候判断下数据有没有发生变化，没发生变化我就会更新成功，发生变化了就不会更新成功我再去重试之前的动作直到更新成功。**

##### ****锁应用****

**悲观锁**

**使用悲观锁的时候我们首先必须关闭mysql数据库的自动提交属性，因为MySQL默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。**

**关闭命令为：set autocommit=0;**

**悲观锁一般使用select…for update实现，在执行的时候会锁定数据，虽然会锁定数据，但是不影响其他事务的普通查询使用。**

**在我们使用悲观锁的时候事务中的语句例如：**

|  |
| --- |
| **//开始事务**  **begin;/begin work;/start transaction; (三选一)**  **//查询信息**  **select \* from order where id=1 for update;**  **//修改信息**  **update order set name='names';**  **//提交事务**  **commit;/commit work;(二选一)** |

**此处的查询语句for update关键字，在事务中只有SELECT ... FOR UPDATE 或LOCK IN SHARE MODE 同一条数据时会等待其它事务结束后才执行，一般的SELECT查询则不受影响。**

**注意事项**

**执行事务时关键字select…for update会锁定数据，防止其他事务更改数据。但是锁定数据也是有规则的。**

**查询条件与锁定范围：**

**1、具体的主键值为查询条件**

**比如查询条件为主键ID=1等等，如果此条数据存在，则锁定当前行数据，如果不存在，则不锁定。**

**2、不具体的主键值为查询条件**

**比如查询条件为主键ID>1等等，此时会锁定整张数据表。**

**3、查询条件中无主键**

**会锁定整张数据表。**

**4、如果查询条件中使用了索引为查询条件**

**明确指定索引并且查到，则锁定整条数据。如果找不到指定索引数据，则不加锁。**

**乐观锁**

**1、使用自增长的整数表示数据版本号，更新时检查版本号是否一致，比如数据库中数据版本为666，更新提交时version=666+1,使用该version值(=667)与数据库version+1(=667)作比较，如果相等，则可以更新，如果不等则有可能其他程序已更新该记录，所以返回错误或者发起重试动作。**

**例如表**

|  |
| --- |
| **student(id,name,version)**  **1 a 1** |

**当事务一进行更新操作：update student set name='txt' where id = #{id} and version = #{version};**

**此时操作完后数据会变为id = 1,name = txt,version = 2，当另外一个事务二同样执行更新操作的时候，却发现version != 1，此时事务二就会操作失败，从而保证了数据的正确性。**

**2、使用时间戳来实现，原理同上。**

**3、使用其他数据库字段，如：金额，更新时添加条件判断金额是否变化，原理同上。**

##### ****结论****

两种锁各有优缺点，不能单纯的定义哪个好于哪个。乐观锁比较适合数据修改比较少，读取比较频繁的场景，即使出现了少量的冲突，这样也省去了大量的锁的开销，故而提高了系统的吞吐量。但是如果经常发生冲突（写数据比较多的情况下），上层应用不不断的retry，这样反而降低了性能，对于这种情况使用悲观锁就更合适。