====2020/09/12====

synchronized volatile lock

一、锁

1. 作用：数据库使用锁是为了支持对共享数据进行并发访问，提供数据的完整性和一致性。
2. 分类：悲观锁、乐观锁、共享锁、排它锁、记录锁、间隙锁、临键锁。

二、锁的具体分类

1. 乐观锁和悲观锁：

a. 乐观锁：

i. 乐观锁是基于乐观的概念，每次去读数据的时候都认为其他事务不会对数据进行修改操作，但是在更新数据的时候，会先判断这个值是否被其他事务修改，如果发生冲突，就进行回滚，cas就是一种乐观锁；

ii. 优点：避免了加锁解锁的开销；

iii. 缺点：增加了冲突。

iv. 乐观锁是在提交的时候检验冲突，悲观锁是加锁避免冲突。资源竞争小，用乐观锁；资源竞争大，用悲观锁。

b. 悲观锁：

i. 认为数据随时会修改，所以在数据处理过程中需要对数据加锁，这样别的事务拿到这个数据就会block阻塞，直到它拿到锁。关系型数据库中的行锁、表锁、读锁、写锁都是悲观锁，在读之前先加上锁；悲观锁的实现方式，依赖于数据库；

ii. 优点：使用阻塞的方式，所以避免冲突；

iii. 缺点：并发性能不好，因为未获得锁会阻塞。

2. 共享锁和排它锁：（按锁的类型分类）

a. 共享锁：

i. 共享锁，S锁，也是读锁，事务A对数据加了共享锁，其他事务也只能对这个数据加S锁，多个用户可以同时读，但不允许有写操作，直到除了某个事务自身外，其他事务都放掉共享锁，这个事务才能获得排他锁。

b. 排它锁：

i. 排它锁，X锁，事务A对对象加X锁以后，只有事务A可以读写该对象，其他事务不能对该对象加任何锁，直到事务A释放X锁。

c. 意向锁：

i. 意向共享锁、意向排它锁；

ii. 意向锁是用来表示事务接下来想要获得的锁的类型；

iii. 意向排它锁和共享锁是不兼容的。

3. 一致性非锁定读和一致性锁定读

a. 非锁定读：

i. 通过MVCC实现，读取的是快照版本。

b. 锁定读：

i. 用法：s锁lock in share mode, x锁for update；

ii. 防止死锁。因为使用共享锁的时候，修改操作，需要先获得共享锁，读取数据，再升级为排它锁，然后进行修改操作。这样如果同时有多个事务对同一个数据对象申请共享锁，然后再同时升级为排它

锁，这些事务都不会释放共享锁，而是等待其他事务释放，就造成了死锁。

iii. 因此，在数据修改前的select 语句中直接申请排它锁，其他数据就无法获取共享锁和排它锁，避免死锁。

MVCC

i. MVCC多版本并发控制，查询需要对资源加共享锁（s），修改需要对数据加排他锁（X）；

ii. 利用undo log回滚日志使读写不阻塞，实现了可重复读。当一个事务正在对一条数据进行修改的时候，该资源会被加上排它锁。在事务未提交时，对加锁资源进行的读操作，读操做无法读到被锁资源，会通过一些特殊的标识符去读undo log 中的数据，这样读到的是事务执行之前的数据。

4. 外键和锁

a. 在Innodb中，对于一个外键，如果没有显示的加索引，会自动地加上索引，避免表锁；

b. 对于外键的修改，需要对父表加S锁。这是，如果父表有X锁，则子表上加S锁的操作会阻塞；如果对父表加上S锁之后，则父表上不会加上X锁，保证了数据的一致性。

5. 行锁的三种算法（记录锁、间隙锁、临键锁）

a. 在可重复读的事务隔离级别下解决幻读的问题。幻读：可重复读的事务隔离级别下可能出现幻读，因为，可重复度保证了当前事务不会读取到其他事务已提交的update 操作。但同时，也会导致当前事务无法感知到其他事务的Insert或delete操作，这就产生了幻读。