MySQL事务、锁

事务

1. 事务概念

数据库事务是数据库执行过程中的一个逻辑单位，一个事务通常包含了对数据库的读/写操作。它的存在包含有以下两个目的：

- 为数据库操作序列提供了一个回滚的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。
- 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法(版面问题下次讨论)，以防止彼此的操作互相干扰。

2. 事务的特性

1. 原子性（aotmic）
   事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。
2. 一致性（consistent）
   事务在完成时，必须使所有的数据都保持一致状态。
3. 隔离性（isolaton）
   由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状 态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据
4. 持久性（duration）
   事务完成后，它对系统的影响是永久性的

锁

1.共享锁、排他锁

1. 共享锁（shared locks,S锁），共享锁又叫读锁，如果事务T1对行R加上S锁，则

- 其它事务T2/T3/Tn只能对行R再加S锁，不能加其它锁
- 获得S锁的事务只能读数据，不能写数据。
  语法：

```
select … lock in share mode;
```

2. 排它锁（exclusive locks,X锁），排它锁又叫写锁，如果事务T1对行R加上X锁，则

- 其它事务T2/T3/Tn都不能对行R加任何类型的锁，直到T1事务在行R上的X锁释放。
- 获得X锁的事务既能读数据，又能写数据（也可以删除数据）。
  语法：

```
select ... for update;
```

举例：

```
// start T1
SELECT * FROM USER WHERE id = 1 lock in share mode; (S锁)

// start T2
UPDATE USER SET name = '小明' WHERE id = 1;

// start T3
// 此时，如果 T3 做同样查询，可以直接获取S锁进行查询
SELECT * FROM USER WHERE id = 1 lock in share mode; (S锁)

//这个时候如果T1（事务1）要进行 DELETE 操作
// start T1
SELECT * FROM USER WHERE id = 1 lock in share mode; (S锁)
DELETE FROM USER WHERE id = 1;
```

如果T1不进行提交，则S锁不会释放，那么T2就拿着X锁眼巴巴的看着，一直等待T1（事务1）释放S锁。
此时，T1发现X锁被T2占据着，所以T1拿不到X锁一直等待T2释放X锁，而T2拿着X锁等待T1释放S锁，这样互相等待就产生了死锁，deadLock。 发生死锁以后，InnoDB 会产生错误信息，并且释放锁。

时间流程：

---	T1(transaction1)	T2(transaction2)	T3(transaction3)
time1	select id=1 加S锁		
time2		update id=1加X锁(等于T1释放S锁)	
time3		等待T1释放...	select id=1(可以加S锁)
time4	delete id=1(等待T2释放X锁)	等待T1释放...	
time5	等待T2释...		
time6	等待T2释...		

2.意向锁

意向锁（Intention Locks）是表锁，多用在innoDB中，是数据库自身的行为，不需要人工干预，在事务结束后会自行解除。
意向锁分为意向共享锁(IS锁)和意向排它锁(IX锁)

- 锁：表示事务中将要对某些行加S锁
- IX锁：表示事务中将要对某些行加X锁

意向锁的主要作用是提升存储引擎性能，innoDB中的S锁和X锁是行锁，每当事务到来时，存储引擎需要遍历所有行的锁持有情况，性能较低，因此引入意向锁，检查行锁前先检查意向锁是否存在，如果存在则阻塞线程。
使用举例：

```
T1:
SELECT * FROM A WHERE id = 1 lock in share mode;  (加S锁)

T2:
SELECT * FROM A WHERE id > 0 for update; (加X锁)
```

看上面这2个SQL事务，T1执行时候，对id=1这行加上了S锁，T2执行前，需要获取全表的更新锁进行判断，即： step1：判断表A是否有表级锁 step2：判断表A每一行是否有行级锁 当数据量较大时候，step2这种判断极其低效。

意向锁协议

- 事务要获取表A某些行的S锁必须要获取表A的IS锁
- 事务要获取表A某些行的X锁必须要获取表A的IX锁

这个时候step2就改变成了对意向锁的判断
step2：发现表A有IS锁，说明表肯定有行级的S锁，因此，T2申请X锁阻塞等待，不需要判断全表，判断效率极大提高

3.间隙锁

当我们用范围条件条件检索数据（非聚簇索引、非唯一索引），并请求共享或排他锁时，InnoDB会给符合条件的数据记录的索引项加锁；对于键值在条件范围内但并不存在的记录，称为间隙，InnoDB也会为这些间隙加锁，即间隙锁。 Next-Key锁是符合条件的行锁加上间隙锁

间隙锁产生的条件：
在InnoDB下，间隙锁的产生需要满足三个条件：隔离级别为RR、当前读、查询条件能够走到索引

间隙锁的作用： 在RR模式的InnoDB中，间隙锁能起到两个作用

- 保障数据的恢复和复制
- 防止幻读 防止在间隙中执行insert语句；防止将已有数据update到间隙中