表级：

只要操作/读行数据，就会有x/s级的意向锁

行级：

**记录锁**：s型锁（共享锁），x型锁（排它锁）🡪不允许两个不同的事务持有同一个行数据的x型锁/s型+x型共存🡪但是s型锁是可以让不同事务持有的

**间隙锁**：(2,5) 【**允许不同事务得到相同/有包含关系的间隙锁🡪但其s型锁和x型锁其实是有意义的，主要是在update/delete操作获取间隙锁时；以及insert操作添加插入意向锁时**】🡪**因为**间隙锁只是为了阻止“其他事务”在这个区间插入/修改行数据，

**Next-key-lock**（临键锁）：[2,5)，(2,5]，[2,5] 之类的，包含了“间隙锁”+“记录锁”【**也分x型和s型；不允许x型的临键锁进行共享🡪会阻塞等待其他事务释放**】

Select \* from xxx; 普通select语句**不加任何锁**

Select \* from xxx where xxx lock in share mode; 可能上记录锁(等值查询)/间隙锁(查询一个不存在的值)/临键锁（比如你进行范围查询）🡪**会上S型的锁**，除了间隙锁，是不允许和X型锁共存的🡪什么时候上什么类型的锁没有研究特别透彻，MySQL会自己选择

Select \* from xxx where xxx for update;/update/delete等语句 一样可能上三种类型的锁🡪**但是是x型的锁**；

**Insert语句**比较特殊，分两种情况：

插入的区间已经有间隙锁，例如：存在**主键索引**的x型间隙锁(2,5)，想要插入id=3，此时会产生“插入意向锁”【虽然是意向锁，但是行级】，阻塞等待间隙锁释放

**唯一键冲突的情况**（比如二级唯一索引/主键索引），分两种情况：

1. 事务进行insert时发现**唯一键重复/主键重复**，在事务结束前，会上s型的“临键锁”(唯一索引唯一键重复的情况)/“记录锁”(主键重复的情况)
2. **两个事务**进行同一个insert语句时，事务a进行插入时只会上“表级锁”；事务b进行插入时，事务a**才**会上x型的“临键锁”/“记录锁”🡪事务b还会尝试添加s型的锁，但是因为存在x型的，只能阻塞等待

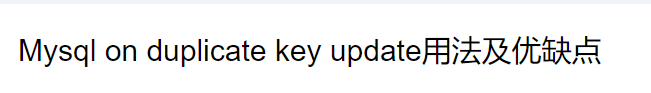
下面的 **insert on duplicate key update**的语句，一般用于插入 有唯一键的行数据 的情况，能实现“**唯一键存在则进行update操作；不存在的情况则进行insert操作**”

假如有两个事务同时执行下面两个语句（唯一键相同的插入语句），其过程类似于：事务a进行 **插入1 操作**，此时只会**上表锁**；事务b进行 **插入2操作**，此时**事务a才会给创建的新行数据上x型的记录锁【实践出来的值】**🡪与两个insert不同的是，此时事务b上的是x型的“临键锁”，而不是s型的“记录锁”

**缺点**：自增主键的速度会加快（但是就算这样也能用几年）；MySQL主从同步时可能会出现主键错误🡪但以上两点都能通过调整一个参数规避

insert into read\_history (bookId,userId,bookName,chapterNum,chapterName,ifCollection) values (16002,110,"插入1",2,"插入1",0) on duplicate key update chapterNum=5;

insert into read\_history (bookId,userId,bookName,chapterNum,chapterName,ifCollection) values (16002,110,"插入2",null,null,1) on duplicate key update ifCollection=0;

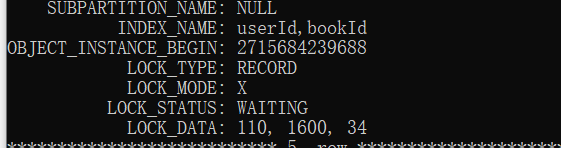


学习的过程：

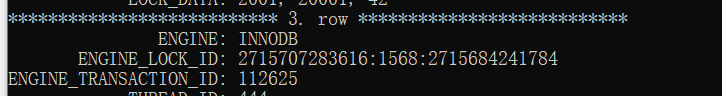
参考文章：小林coding <https://www.xiaolincoding.com/mysql/> 的锁篇

select \* from performance\_schema.data\_locks\G; 通过该指令可以看见此时数据库的锁情况





留意这些部分：目前这个表示 “行级锁”，x型的“临键锁”，正在阻塞获取该锁，锁的数据；



这里的engine\_transaction\_id表示事务id，用于确定某个锁是哪个事务持有/正在阻塞等待获取的