**介绍的是锁的类型：而非具体使用**

**（行级）读锁和排他锁（Shared and Exclusive Locks）**

Shared lock：允许其他锁读和其他锁加读锁，阻塞写

Exclusive lock:阻塞其他事物的读写操作

**（表级）意向锁（intention lock | intention Shared and Exclusive）**

Alter,drop，LOCK TABLES ... WRITE会加表级锁，

表级锁与行锁之间会相互阻塞:有任何行的读写操作是会阻塞相应的表锁，所以需要添加意向锁解决这个问题。

规则：加读锁前要加is(intention share lock)

加写锁前要加ix (intention exclusive lock)

意向锁不阻塞权标操作外的任何请求，

|  | ***X*** | ***IX*** | ***S*** | ***IS*** |
| --- | --- | --- | --- | --- |
| ***X*** | Conflict | Conflict | Conflict | Conflict |
| ***IX*** | Conflict | Compatible | Conflict | Compatible |
| ***S*** | Conflict | Conflict | Compatible | Compatible |
| ***IS*** | Conflict | Compatible | Compatible | Compatible |

//表中的s/x是表级锁

表级读锁与表级写锁/意向写锁相互阻塞

表级写锁与四种锁都互相阻塞（is/ix被阻塞后，之后的行锁自然也无法得到）

**记录锁（record locks）（innodb的行级锁的实现方式）**

单条索引记录上加锁，record lock锁住的永远是索引(lock record)数据依靠索引建的，即使该表上没有任何索引，那么innodb会在后台创建一个隐藏的聚集主键索引，那么锁住的就是这个隐藏的聚集主键索引。所以说当一条sql没有走任何索引时，那么将会在每一条聚集索引后面加锁，这个类似于表锁，但原理上和表锁应该是完全不同的。

Update/delete |SELECT c1 FROM t WHERE c1 = 10 FOR UPDATE|inshare mode会生成record lock

阻塞insert, update, 或delete

**间隔锁\*（Gap Locks）**

Gaplock也是针对索引，只不过是根据区间。

SELECT c1 FROM t WHERE c1 BETWEEN 10 and 20 **FOR UPDATE**;

若c1有唯一索引则阻止其他事物插入10-20之间，也阻止更新/删除区间的数据。

没有则锁全表，同recordlock。

若隔离级别为read-commited，则无效。

防止幻行。

**Next-Key Locks**

Next-Key ：record lock+gap lock

mysql的默认事务隔离级别是可重复读，并且innodb\_locks\_unsafe\_for\_binlog参数为0，这时默认采用next-key locks

若索引值有1，3，5，8，11,则gap lock区间为（-∞,1]，(1,3]，(3,5]，(5,8]，(8,11]，(11,+∞），若扫描到8则锁定的范围是（5,8]，下个下个键值范围是（8,11]，所以插入5~11之间的值的时候都会被锁定。

快照读：

简单的select操作，没有lock in share mode或for update，快照读不会加任何的锁，而且由于[MySQL](http://lib.csdn.net/base/mysql)的一致性非锁定读的机制存在，任何快照读也不会被阻塞。但是如果事务的隔离级别是SERIALIZABLE的话，那么快照读也会被加上共享的next-key锁，本文不对SERIALIZABLE隔离级别做叙述。

gap lock的内部加锁原理（同next-key）

gap lock的前置条件：（或条件）

1 事务隔离级别为REPEATABLE-READ，innodb\_locks\_unsafe\_for\_binlog参数为0，且sql走的索引为非唯一索引

2 事务隔离级别为REPEATABLE-READ，innodb\_locks\_unsafe\_for\_binlog参数为0，且sql是一个范围的当前读操作，这时即使不是非唯一索引也会加gap lock

select .. from (**不被阻塞不阻塞他人，没有争抢没有阻塞**)

快照读不加任何类型的锁 ，走mvcc

select...from lock in share mode 在扫描到的任何索引记录上加共享的（shared）next-key lock，

还有主键聚集索引加排它锁

select..from for update 在扫描到的任何索引记录上加排它的next-key lock，还有主键聚集索引加排它锁

update..where delete from..where 在扫描到的任何索引记录上加next-key lock，还有主键聚集索引加排它锁

insert into..

简单的insert会在insert的行对应的索引记录上加一个排它锁，这是一个record lock，并没有gap，所以并不会阻塞其他session在gap间隙里插入记录。