写锁是一定会排斥写锁的，不然多个事务一定会一起写肯定玩完了。

redolog的存在保证了事务的持久性。

## Read uncommit

读已提交，读已提交的原因在于再这种状况下，对于某列的写事务加了写锁，但是这个写锁并不会排斥读锁，这边写的过程中读事务还是可以继续进行，就导致了脏读。

这种基本是存在可重复读的，这种状态读是不排斥写的，所以这个读事务的过程中，完成了一个写事务，那么读事务前后数据的状态是不一样的。

幻读肯定也存在。

## Read commit

读锁没有排斥写锁的意思就是 我这边事务要读两次,前后两次读操作中间其他事务写了东西,那么前后两次读出来的就不一样了.

这种方式就是写事务排斥的读事务，这行数据的写锁排斥了读锁，也就是这行数据写的过程中不能再读了，那么就不会脏读了，但是这行数据读锁没有排斥写锁，也就是这行数据读的过程中中间进行了一个这行数据的写事务，那么读事务前后的数据就不一致的，就有重复度现象。

## Repeatable read

这个的意思就是这行数据的读锁排斥了写锁，这样的话我读的时候你就不能写了这行数据了。到这里读排斥了写，写也排斥了读，按理说已经完美了，读写，写写之间都互相排斥，只有读读之间可以，那么对这行数据的操作绝对不会出现问题了，确实，这行数据的操作不会出现问题了，但是会有这种情况，一个事务是一个表比如所有数据的某一列的和，这个是不会锁表的，这个读事务确实会锁住所有的当前的数据，但是其他线程不修改当前这写些行的数据，而是插入了一个新的，这样的话是没有排斥的（也就是读操作没有锁表）所以这个读事务前后就变了，那么就出现了幻读。

## Serializable

串行执行，没啥说的。