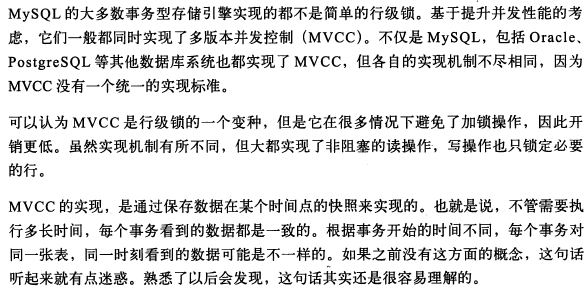
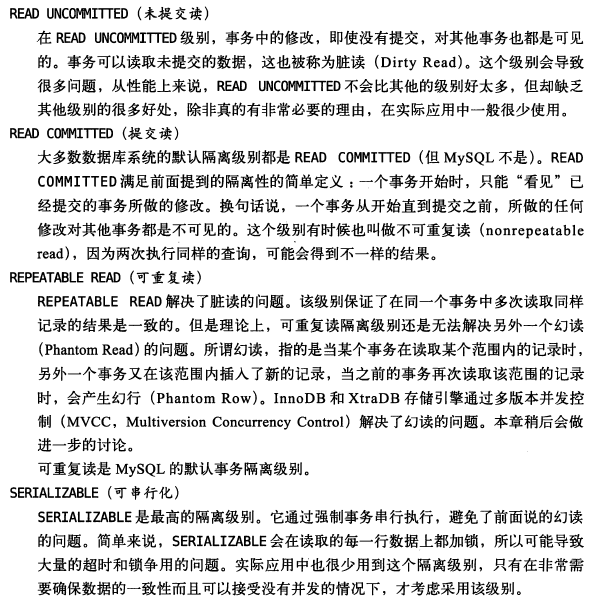
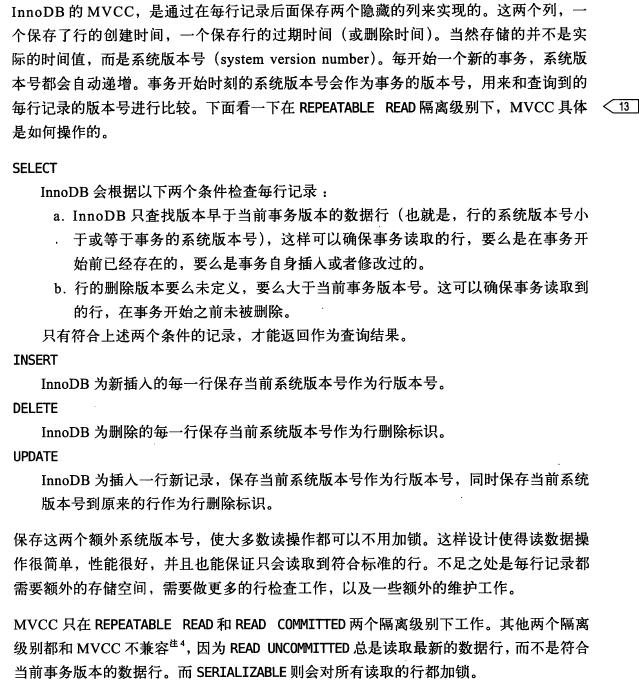
事物隔离级别的实现=锁+mvcc

大多数读->mvcc->非阻塞

Update->锁部分 ->gap lock







问题

读未提交：可读到其他事物未提交写

读已提交：不可读到其他事物未提交写，但在一个读事务内的两次读可能有写事物提交导致两次数据不一致。

可重复读：保证了同一事物两次读同一行一致。（mvcc）但可能两次读中间插入了一行，造成读的行数不一致，造成幻行问题。（mysql默认隔离级别）

串行化：所有操作串行。

基于mvcc，读 不会阻塞写

例:(REPEATED\_READ)

A B

1. Start transaction ;
2. select \* from tblWKTask where id =11; start transaction;

--+---------------+

| banner |

--+---------------+

| www.baidu.com |

--+---------------+

3. update tblWKTask set banner='http' where id =11;

4. select \* from tblWKTask where id =11;

--+---------------+

| banner |

--+---------------+

| www.baidu.com |

--+---------------+

5. commit;

6 select \* from tblWKTask where id =11;

--+---------------+

| banner |

--+---------------+

| www.baidu.com |

--+---------------+

7. Commit;

8. select \* from tblWKTask where id =11;

--+---------------+

| banner |

--+---------------+

| http |

--+---------------+

结论

1. 读不阻塞写
2. 读Mvcc的版本是根据事物开始的时间
3. 写事物的提交时间是mvcc的版本时间

读则就是按照一般论的事务隔离界别阻塞来做

Binary logging not possible. Message: Transaction level 'READ-COMMITTED' in InnoDB is not safe for binlog mode 'STATEMENT'

锁分为三种

1，Record Lock：单个行记录上的锁。

2，Gap Lock：间隙锁，锁定一个范围，但不包括记录本身。GAP锁的目的，是为了防止同一事务的两次当前读，出现幻读的情况。

3，Next-Key Lock：1+2，锁定一个范围，并且锁定记录本身。对于行的查询，都是采用该方法，主要目的是解决幻读的问题。