锁分类：

1. 对数据的操作类型，分为读锁和写锁
2. 对数据的操作粒度，分为表锁和行锁

读锁（共享锁）：针对同一份数据，多个读操作可以同时进行而不会互相影响

写锁（排它锁）：当前写操作没有完成前，会阻断其他的读锁和写锁

表锁：

偏读，偏向MyIsam存储引擎，开销小，加锁快；无死锁；锁定粒度大，发生冲突的概率最高，并发度最低。

手工增加表锁：

lock table 表名 write;

lock table 表名 read;

查看表的加锁情况：show open tables;

手工释放表锁：unlock tables;

栗子：前提 mylock表的存储引擎是myisam

session 1 对mylock表加读锁，此时：

session 1可以读mylock表，不能修改mylock表，无法读其他表（这是为啥？）。

Session 2可以读mylock表，修改mylock表会阻塞，等session 1释放锁后会立马修改。可以查看和修改其他表。

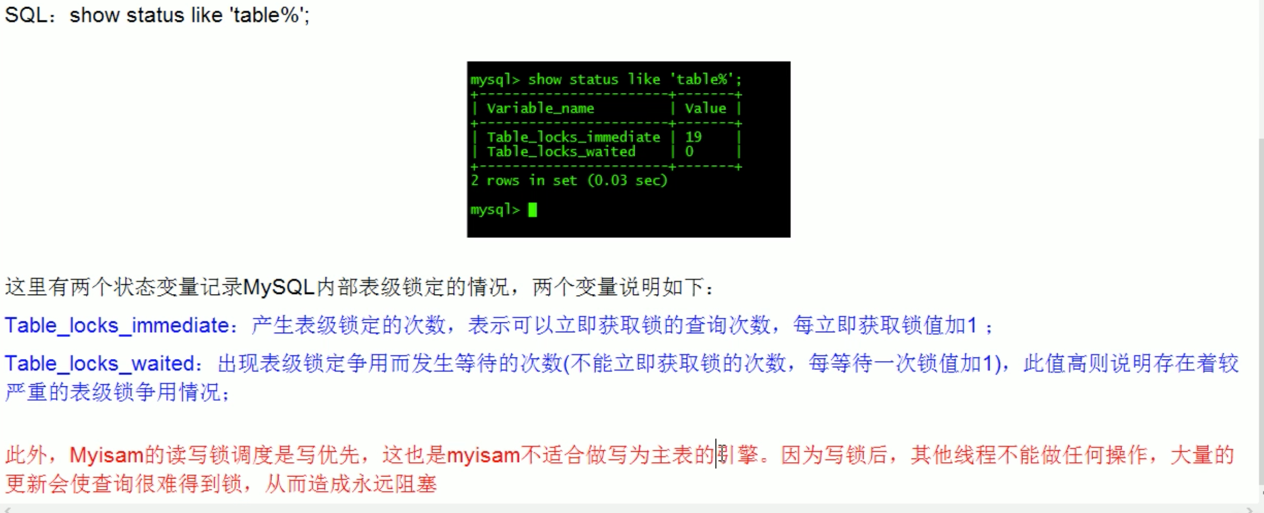
Session 1 对mylock表加写锁，此时：

Session 1可以读mylock表，可以修改mylock表，无法读其他表（同上。为啥？不能读，更加不能修改其他表了）

Session 2 读mylock表会阻塞，等到session 1释放了mylock表，session 2读mylock表会立马返回结果。

总结：读锁会阻塞写，但是不会阻塞读；写锁会阻塞读和写。

查看当前锁状态的命令：



行锁：

偏向Innodb存储引擎，开销大，加锁慢；会出现死锁；锁的粒度小，发生锁重冲突的概率最小，并发最高。

InnoDB与MyIsam最大的不同：

Innodb支持事务；Innodb支持行锁。

事务

隔离级别

行锁演示：

mysql默认会自动提交。

默认不自动提交：set autocommit=0;

栗子看视频吧。。

懒得写了。。。

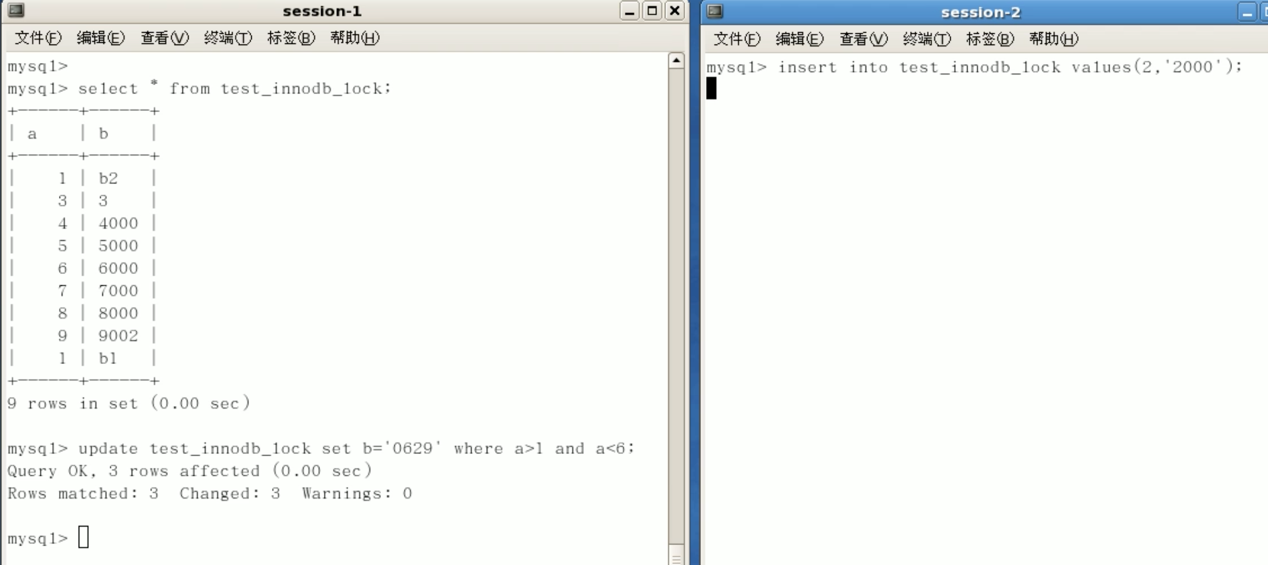
CAP：一致性；高可用性；分区容错性

保CP，保AP

比较危险的一种操作：

如果update时，因为索引失效，可能行锁会变成表锁，影响到整张表的更新。

间隙锁：



上图是间隙锁演示栗子。

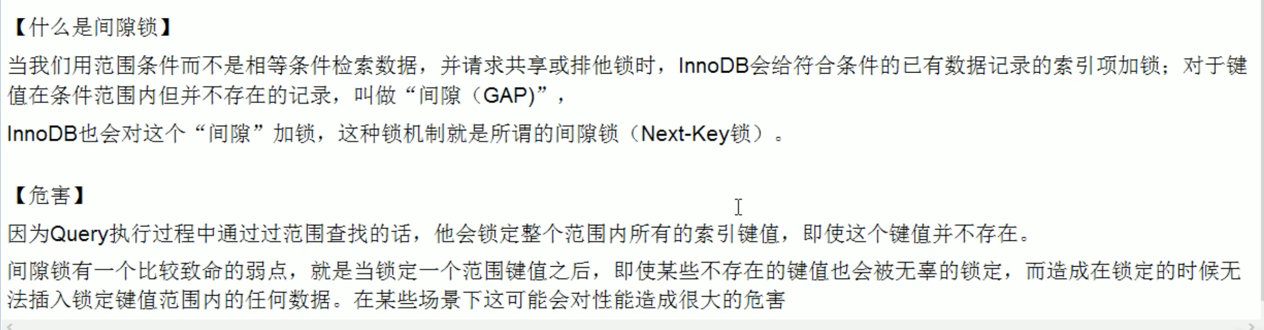
两个会话框中都设成了不自动提交。

Session 1中update的条件是一个范围，执行时会加上间隙锁；session 2中新增的记录正好在间隙锁的范围内，所以insert会阻塞。

等session 1 commit，session 2中的就能自动提交。

最后结果中，两个会话都结束之后，最终的结果是，a=2这行对应的b字段值是2000

间隙锁含义：

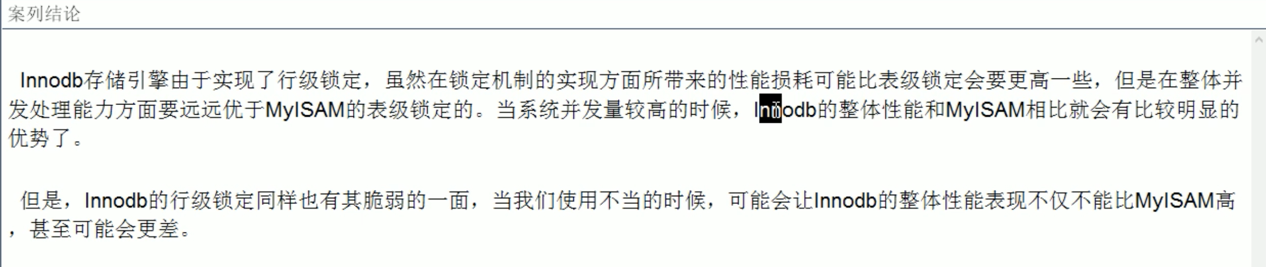


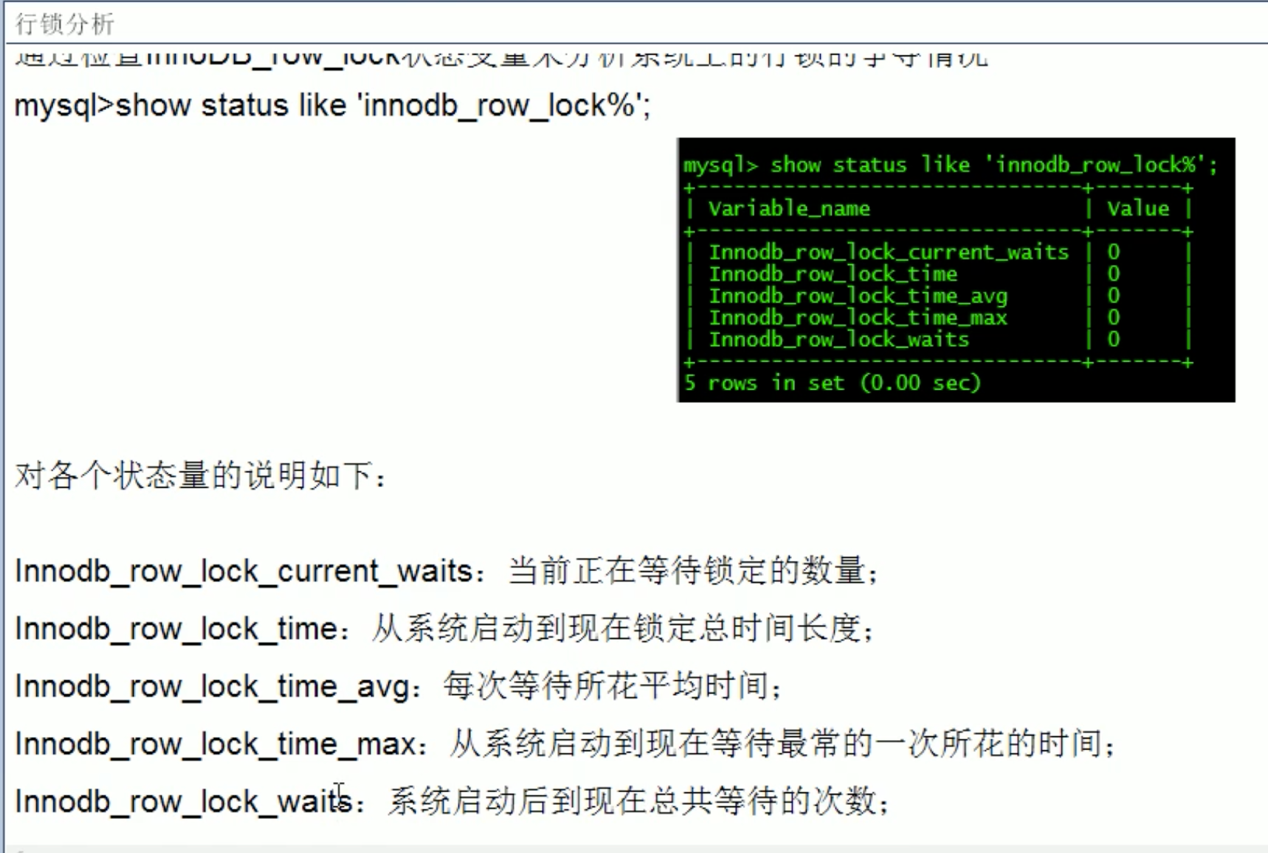
手工加行锁：

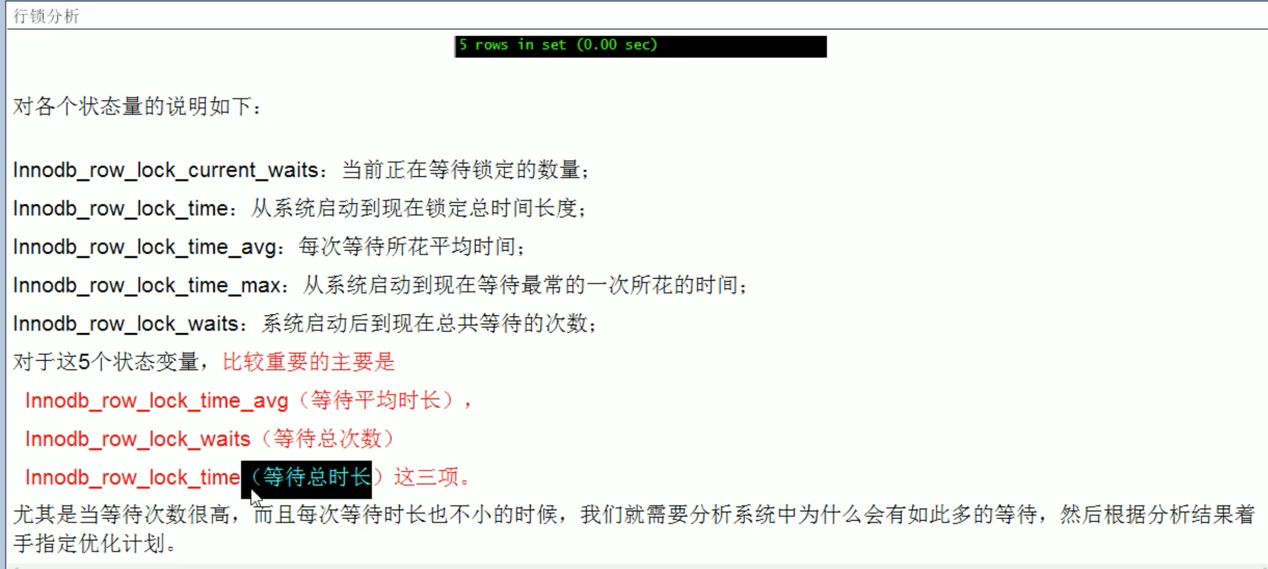


session 1中的 select \* from … **for update**; 可以手动的把满足a=8的记录加上行锁。右边session 2同时在更新a=时，会阻塞住，知道session 1中手动commit

行锁总结：







如果想进一步分析，可以使用show profile; 看日志，分析每个SQL具体的每一步的耗时情况。

优化建议：

1. 尽可能让所有数据检索都通过索引完成，避免无索引行锁升级为表锁
2. 合理设计索引，尽量缩小锁的范围
3. 尽可能减少检索条件，避免间隙锁
4. 尽量控制事务大小，减少锁定资源量和时间长度
5. 尽可能低级别事务隔离

页锁：

介于行锁和表锁之间，会出现死锁，并发量一般。