数据库的隔离级别

# 事物的ACID属性

### 原子性：Atomicity

### 一致性：Consistency

### 隔离性：Isolation

### 持久性：Durability

# MySQL事物的4种隔离级别

### 读未提交Read-Uncommitted

### 读已提交 Read-Committed

### 可重复读 Repeatable-Read

### 可串行化 Serializable (使用很少)

4种现象：

### 更新丢失：

两个事物A和B同时操作相同数据，后提交的事物会覆盖先提交的事物的处理结果，通过乐观锁就可以解决。

### 脏读 Dirty Read：

事物A读取了事物B已经修改但是**尚未提交的数据**；如果事物B回滚，则事物A读取的数据无效，不符合一致性。

### 不可重复读NonRepeatable Read：

事物A读取到了事物B已经提交的修改数据，不符合隔离性。

对应的“可重复读”:事物A无法读取其他事物提交的数据，只能读取事物A开始之前的数据。

### 幻读 Phantom Read：强调数据有新增加。

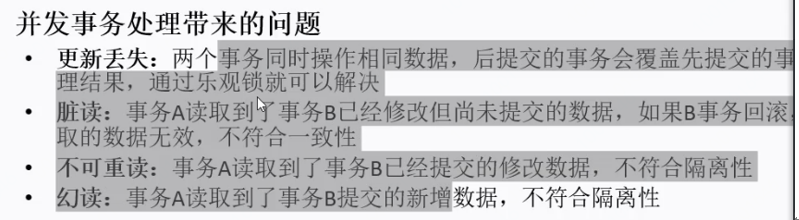
事物A读取到了事物B提交的**新增数据**，不符合隔离性。



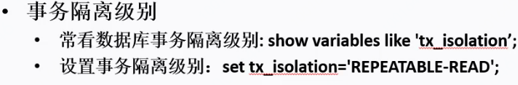
读未提交==>脏读；

读已提交==>不可重复读；

可重复读==>幻读，读到新增的数据。



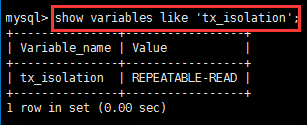
# MySQL查看与设置事物隔离级别



## 查看隔离级别

查看MySQL数据库隔离级别命令：

**show variables like 'tx\_isolation';**



**MySQL数据库事物的默认隔离级别：REPEATABLE-READ可重复读**

**其他大部分数据库的事物默认隔离级别是Read-Committed读已提交。**

## 设置事物隔离级别

set tx\_isolation='**Read-Uncommitted**';

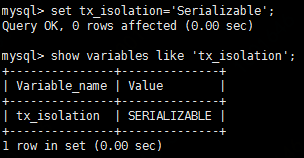
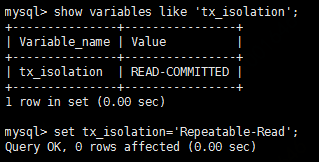
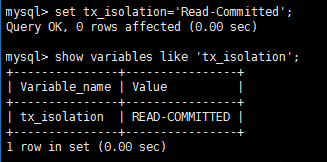
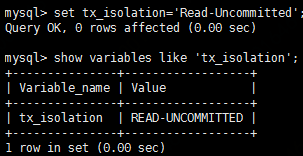
set tx\_isolation='**Read-Committed**';

set tx\_isolation='**Repeatable-Read**';

set tx\_isolation=**'Serializable'**;

命令：

**set tx\_isolation='REPEATABLE-READ';**



# 演练事物的隔离级别及现象出现

## 事物开启、提交、回滚

命令：begin

commit

rollback

## 模拟读未提交Read-Uncommitted：

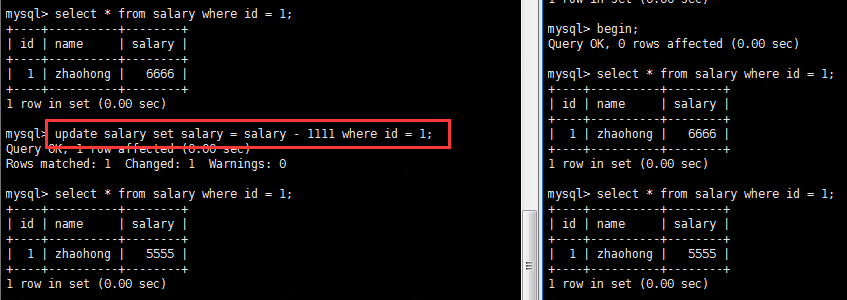
首先两端都设置事物隔离级别为Read-Uncommitted

**set tx\_isolation = 'Read-Uncommitted';**

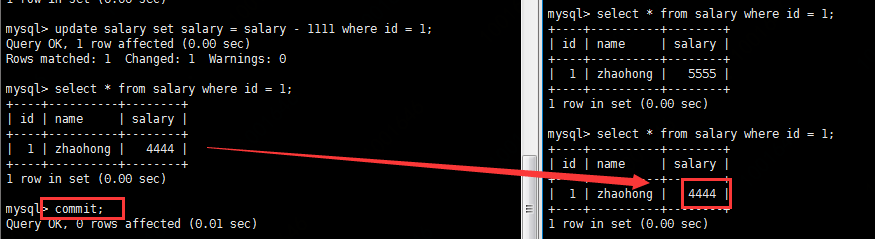
**注意：更新数据直接：**

**update salary set salary = salary - 1111 where id = 1;**

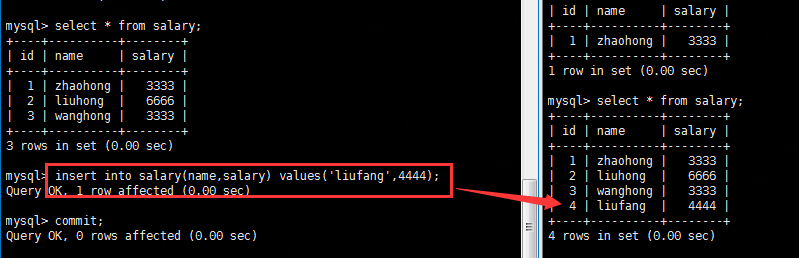
**不要先读取数据到变量中，然后处理后，再更新。**



能够读到**未提交的数据**，当然更能读到已提交的数据，出现**不可重复读现象**。

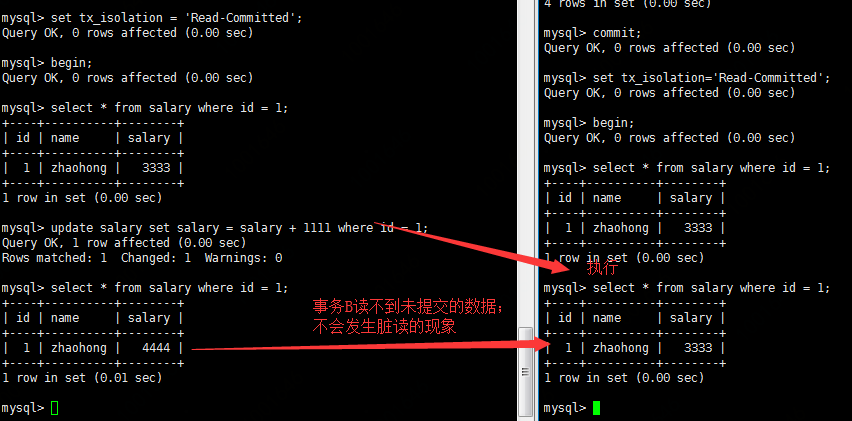


事物B读到了事物A中新增加的数据，幻读：

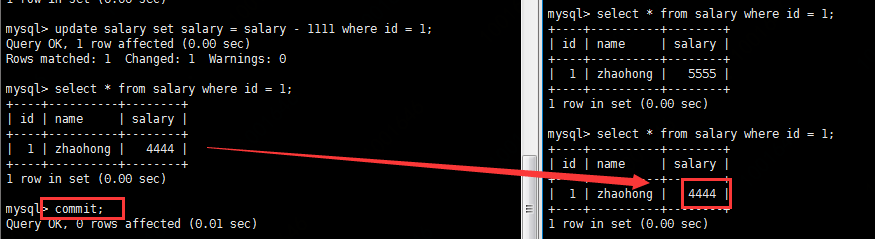


## 模拟读已提交Read-Committed

### 不会出现脏读的情况:

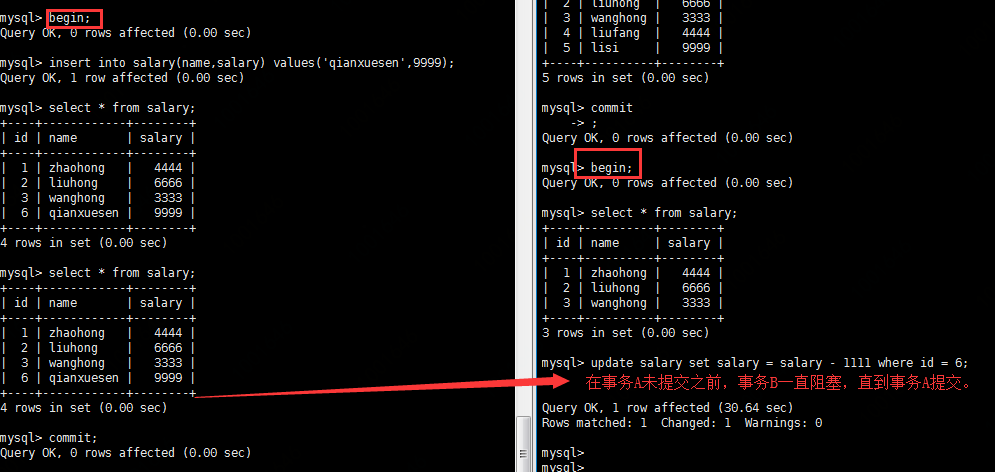


### 有可能出现不可重复读现象：



### 可能出现幻读

若事物A未提交，事物B读取不到事物A新增加的数据，若事物B试图更新事物A新增加但未提交的数据，会一直阻塞，直到事务A提交数据。

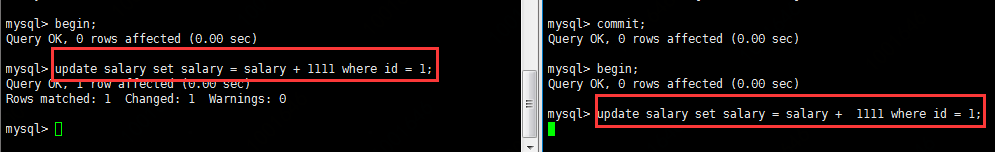


在**读已提交**隔离级别下：

阻塞：

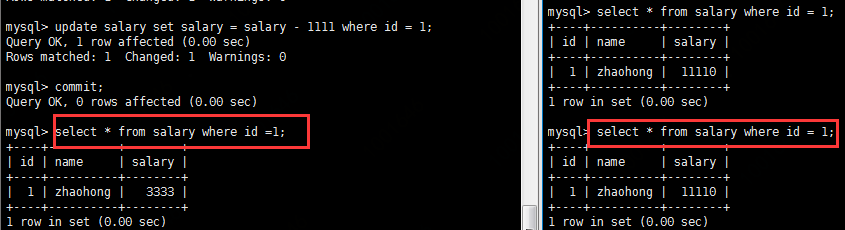
两个事物同时操作同一条记录时，先操作的成功执行，另一个会一直阻塞，直到先操作的事物提交。

幻读问题：**虽然满足可重复读问题，但是事物A增加一个新记录，事物B虽读取不到事物A新增加的记录，但是试图修改该记录时，会一直阻塞**。(相当于其实读到了，但是假装没有读到，真正操作该记录时，会一直阻塞等待)

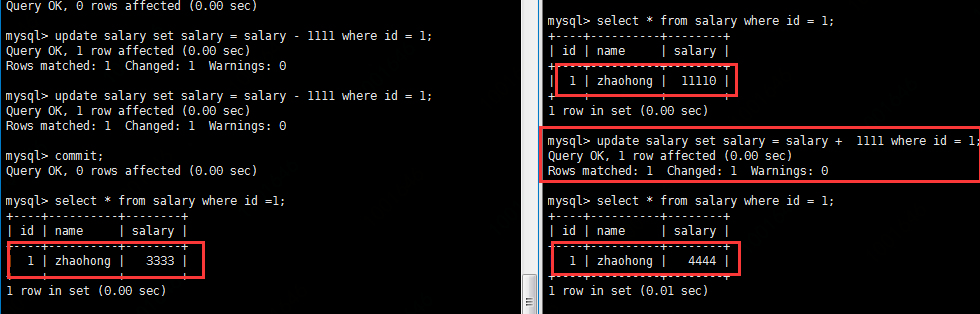


## 模拟可重复读Repeatable-Read

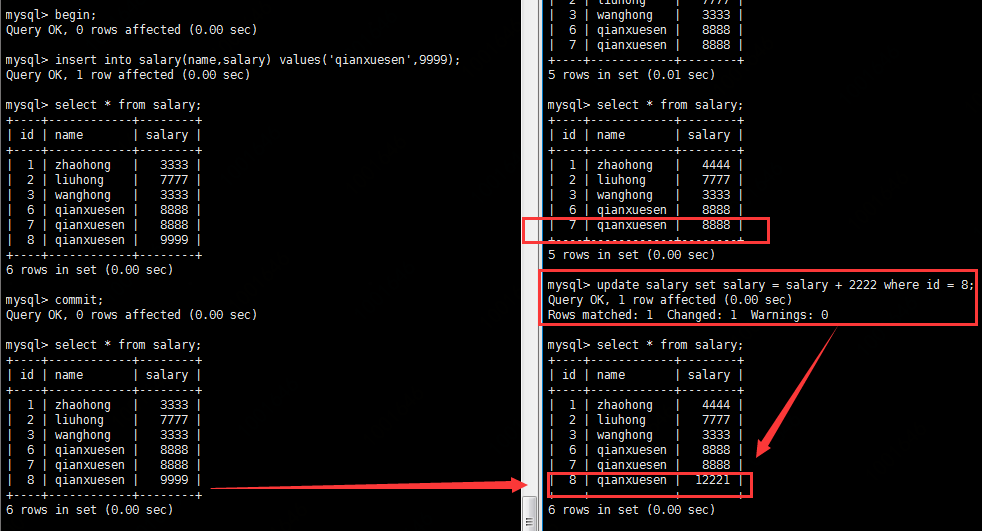
无论其他事务怎么对数据修改，原事务每一次读取的都是最初事物开始的数据。



读的数据始终就是起初读到的数据，但是，若对修改的数据修改，就会在最新的基础上修改。



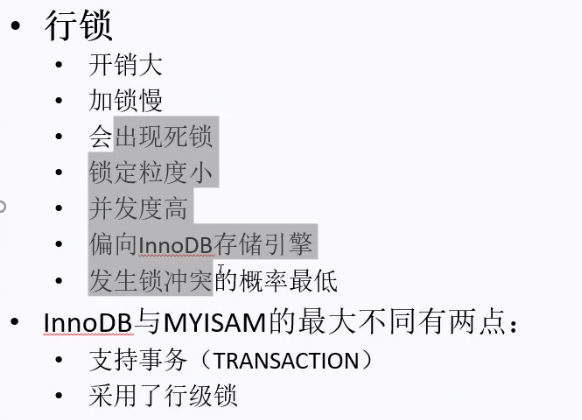
幻读：

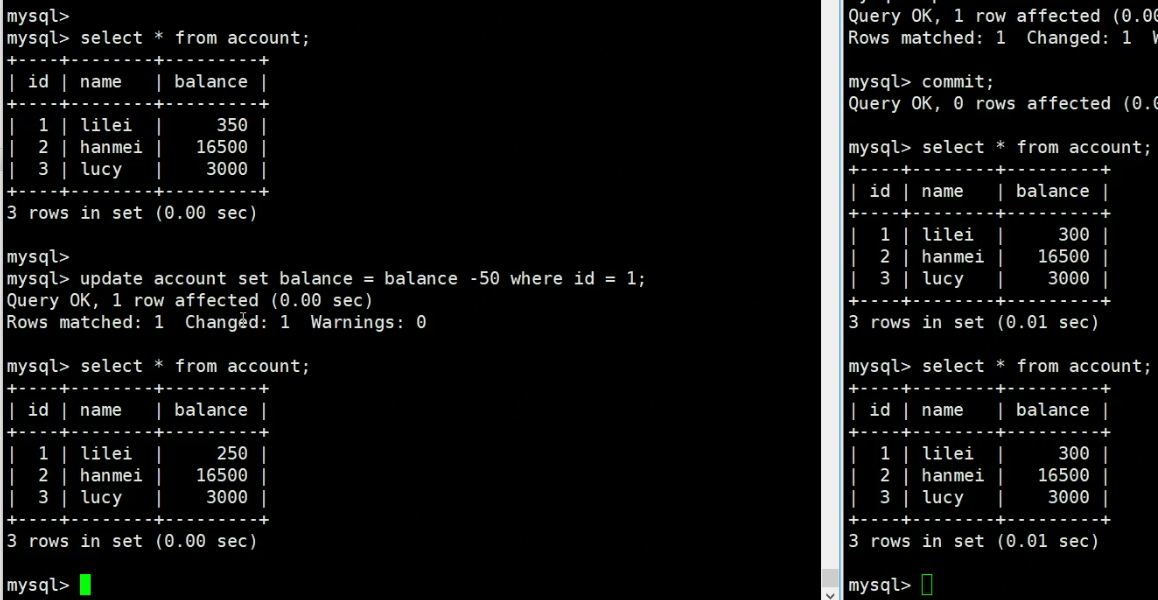


<https://blog.csdn.net/hjy930226173/article/details/79445737>



开启一个事物：





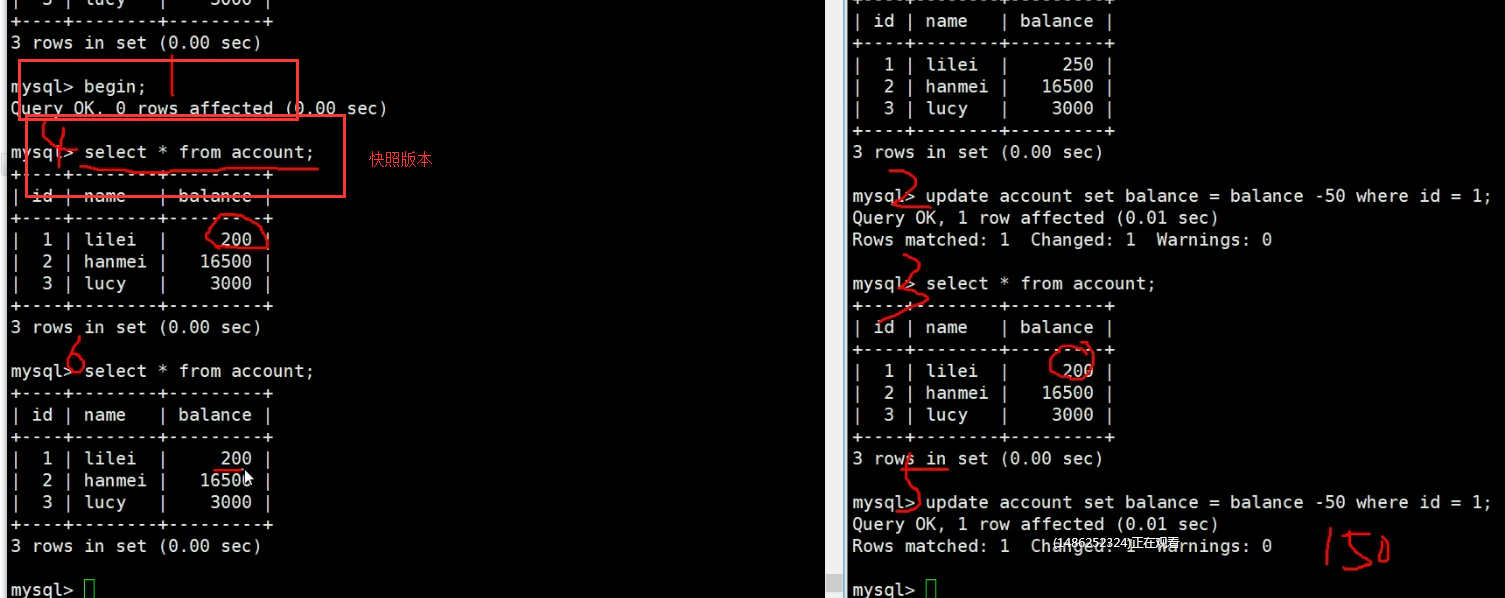
MVCC机制

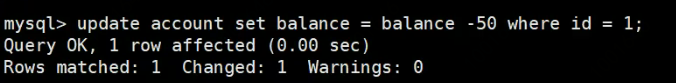
多版本并发控制

没开启一个事物，一条语句就是一个事物。

读的话是按照快照版本；

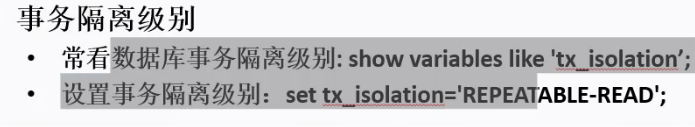
但是更新的话是从数据库中读取最新的值，再修改。

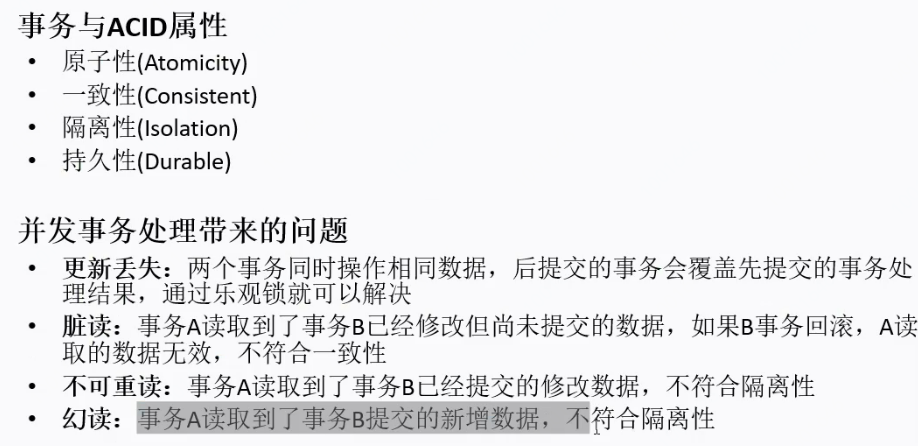




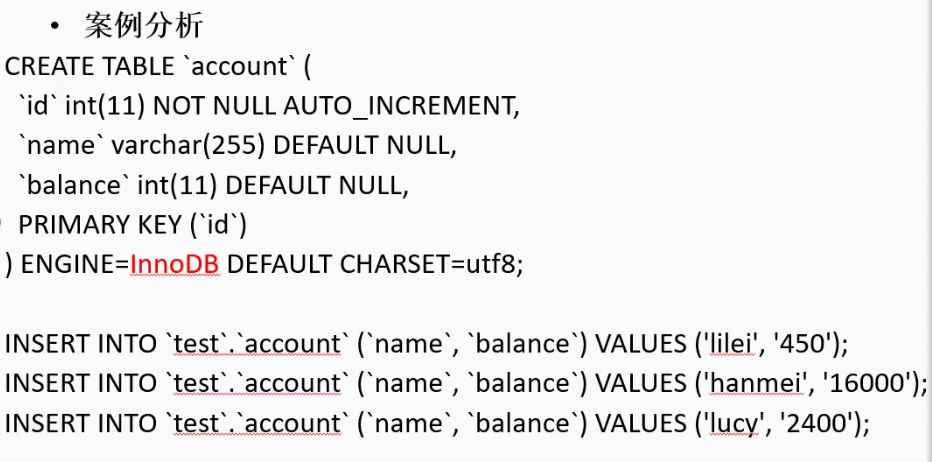
更新数据采用这种语句，不要采用先读取值到变量，然后再根据读到的值更新。

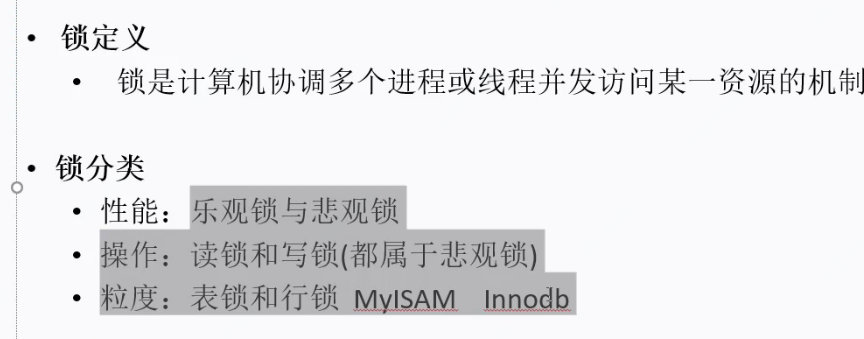
直接将当前值减一。













间隙锁

’