# .锁的概述

## 定义

锁是计算机协调多个进程或线程并发访问某一资源的机制。

在数据库中，除传统的计算资源（如CPU、RAM、I/O等）的争用以外，数据也是一种供许多用户共享的资源。如何保证数据并发访问的一致性、有效性是所有数据库必须解决的一个问题，锁冲突也是影响数据库并发访问性能的一个重要因素。从这个角度来说，锁对数据库而言显得尤其重要，也更为复杂。

## 锁的分类

### 对数据操作的类型（读\写）分

#### 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

当其他写操作没有完成前，它会阻断其他的写锁和读锁。

### 从对数据操作的粒度分

#### 行锁

#### 表锁

# .三锁

开销、加锁速度、死锁、粒度、并发性能

只能就具体应用的特点来说哪种锁更合适。

## \*表锁（偏读）

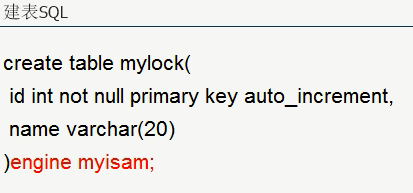
### 特点

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

### 使用

#### 建表

**建表的时候要写上使用MyISAM引擎**



#### 手动增加锁

Lock table 表名字 read**|**write,表名字2 read**|**write,其他;

#### 查看表上加过的锁

Show open tables;

#### 解锁

Unlock tables; 解除库中所有表锁

### 读写锁

#### 读锁

Lock table 表名字 read,表名字2 read**;**

设置读锁后

1. 被锁的表可以被本连接和其他连接读取
2. 其他连接可以读其他表，不受影响
3. 本连接不能读其他表，只能读自己锁了的表。
4. 本连接不能写操作，会报错
5. 其他连接如果进行写操作，会被堵塞，一直等待锁被释放。

#### 写锁

Lock table 表名字 write,表名字2 write**;**

设置写锁后

1. 加锁的表本连接可以读
2. 加锁的表其他连接如果读，被阻塞，一直等待锁被释放
3. 本连接不可以读其他表
4. 本连接可以写加锁的表

### 总结

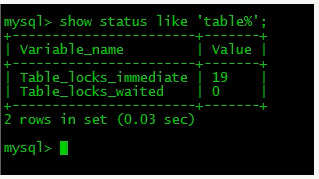
**简而言之，读锁会阻塞写，不会阻塞读，写锁全部阻塞。**

**此外，Myisam的读写锁调度室写优先，这也是myisam不适合做写为主表的引擎。因为写锁后，其他线程不能做任何操作，大量的更新会使查询很难得到锁，从而造成永远阻塞。**

#### 如何分析表锁定

可以通过检查table\_locks\_waited和Table\_locks\_immendiate状态变量来分析系统上的表锁定

SQL：show status like ‘table%’;



**Table\_locks\_immediate：产生表级锁定的次数，表示可以立即获取锁的查询次数，每立即获取锁，值加1；**

**Table\_locks\_waited：出现表级锁定争用而发生等待的次数（不能立即获取锁的次数，每等待一次锁值加1），此值高表示存在着严重的表级锁争用情况。**

## \*行锁（偏写）

### 特点

偏向innodb存储引擎，开销大，加锁慢，会出现死锁，锁定粒度最小， 发生锁冲突的概率最低，并发度也最高。

Innodb与MyISAM的最大不同有两点：1支持事务；2.采用行锁

### 事务

### 行锁案例解释

使用innodb数据库引擎，先把默认提交关闭，autocommit=0，事务，然后在两个连接中操作，默认的隔离级别是可重复读。

连接A，连接B

1. A连接修改一条数据，未提交

A连接读取该数据是修改后的值，读己之所改

B连接会读到修改之前的数据，不会出现脏读。

1. A连接修改一条数据，未提交，B也修改这条数据

B连接会被阻塞。

1. A连接修改一条数据，B连接修改其他数据

各不干扰，都能执行成功。

### 无索引，行锁升级为表锁

**无索引代表没有创建索引或者索引失效。修改表时，就会锁整张表。**

### 间隙锁

#### 解释

当我们用范围条件而不是相等条件检索数据，并请求共享或排他锁时，innoDB会给符合条件的已有数据记录的索引项加锁；对于键值**在条件范围为但是并不存在的记录**，叫做“间隙”。

#### 举例

更新id>1 and id<7的值，但未提交，id 1-7就会被锁起来，这个时候如果有人添加id在范围中的记录，会阻塞，修改范围中的记录也会被阻塞。

#### 危害

因为query执行过程中通过范围查找的话，他会锁定整个范围内所有的索引键值，即使这个键值不存在。

间隙锁有一个比较致命的弱点，就是当锁定一个范围之后，即使某些不存在的键值也会被无辜的锁定，而造成在锁定的时候无法插入锁定键值范围内的任何数据。在某些场景下这可能会对性能造成很大的危害。

### 面试题，如何锁定具体的某一行

这里使用手动上锁

Begin;



执行以上SQL，在未提交本次操作之前，本行会被锁定，造成其他连接的阻塞。

Commit;

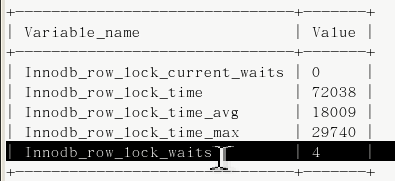
### 总结

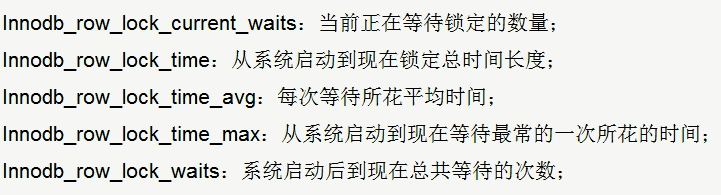
Innodb存储引擎由于实现了行级锁定，虽然在锁机制的实现方面所带来的性能损耗可能比表级锁定会要高一些，但是在整体并发处理能力方面要远远由于MyISAM的表级锁定的。当系统并发量较高的时候，Innodb的整体性能和MyISAM相比就会有比较明显的优势了。

但是，Innodb的行级锁定同样也有其脆弱的一面，当我们使用不当的时候，可能会让Innodb的整体性能表现不仅不能比MyISAM高，甚至会更差。

### 查看行锁

Show status like ‘innodb\_row\_lock%’;





### 优化建议

1. 尽可能让所有数据检索都通过索引来完成，避免无索引行锁升级为表锁。
2. 合理设计索引，尽量缩小表的范围
3. 尽可能较少的检索条件，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（了解）

