|  |  |
| --- | --- |
| 读锁（共享锁）read | |
| A会话 | B会话 |
| Use 库 | Use 库 |
| 加读锁  Lock table jobs read |  |
| Select √ | Select √ |
| Update ×  Insert ×  Delete × |  |
| 读其他表 × | 读其他表 √  更新其他表 √ |
|  | 对jobs表  Insert、delete、update  阻塞 |
| 解锁，释放锁  Unlock tables |  |
|  | 获得锁，Insert或update操作完成（等待了一段时间，系统性能低） |

表级锁

|  |  |
| --- | --- |
| 写锁（排他锁，独占锁）write | |
| A会话 | B会话 |
| Lock table jobs write |  |
| Select √ | Select 阻塞 |
| Delete √  Update √  Insert √ | Delete 阻塞  Update 阻塞  Insert 阻塞 |
| 操作其他表 × | 查询其他表 √ |
| 解锁 | 阻塞得到执行 |

**结论：读锁会阻塞写，不会阻塞读；写锁会阻塞读和写。**

**因为写锁会造成大量阻塞，表锁偏读。**

**相关命令：**

**Show open tables；**

**Lock table jobs read；**

**Unlock tables；**

**Lock table jobs write；**

**Unlock tables；**

**show status like ‘table%’;**

**table\_locks\_immediate**

**table\_locks\_waited:出现表级锁定争用而发生等待的次数**

行锁

|  |  |
| --- | --- |
| 行锁定 | |
| A会话 | B会话 |
| Set autocommit=0 | Set autocommit=0 |
| Update |  |
| Select 新数据 | Select 原始数据 |
| Commit | Select 原始数据 |
|  | Commit |
|  | Select 新数据 |
|  |  |
| 给某一行加锁 | 更新同一行记录 阻塞 |
| Commit | 阻塞操作 完成 |
|  | Commit |
| 查询 | 查询 |
|  |  |
| 给某一行加锁 | 更新另一条记录  成功 |
| Commit | Commit |
| 查询 数据统一 | 查询 数据统一 |

如何锁定一行

A会话 B 会话

Select \* from jobs Update jobs set min\_salary=0

where job\_id=’Sh\_clerk’; where job\_id=’SH\_clerk’

for update ; 阻塞

commit 更新成功

commit

查询（数据统一） 查询（数据统一）

**间隙锁**

A会话 B会话

Update 。set 。where a>1 and a<6 insert..(2,..) 阻塞

Commit 插入成功

Commit

新数据（包含更新和插入2） 新数据（包含更新和插入2）

当用范围条件而不是相等条件检索数据，并请求共享或者排他锁时，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内但并不存在的记录，叫做‘间隙GAP’。

Innodb也会对这个间隙加锁，这种锁机制就是间隙锁。

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值，即使这个键值不存在。间隙锁有一个致命的弱点，就是当锁定一个范围键值之后，即使某些不存在的键值也会被无辜锁定，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。某些场景下会对性能造成很大的危害。