### 1、前言

MySQL常用引擎有MyISAM和InnoDB，而InnoDB是mysql默认的引擎。MyISAM不支持行锁，而InnoDB支持行锁和表锁。

如何加锁？

MyISAM在执行查询语句（SELECT）前，会自动给涉及的所有表加读锁，在执行更新操作（UPDATE、DELETE、INSERT等）前，会自动给涉及的表加写锁，这个过程并不需要用户干预，因此用户一般不需要直接用LOCK TABLE命令给MyISAM表显式加锁。

显式加锁：

上共享锁（读锁）的写法：lock in share mode，例如：

select math from zje where math>60 lock in share mode；

上排它锁（写锁）的写法：for update，例如：

select math from zje where math >60 for update；

### 2、表锁

不会出现死锁，发生锁冲突几率高，并发低。

MyISAM引擎

MyISAM在执行查询语句（select）前，会自动给涉及的所有表加读锁，在执行增删改操作前，会自动给涉及的表加写锁。

MySQL的表级锁有两种模式：

* 表共享读锁
* 表独占写锁

读锁会阻塞写，写锁会阻塞读和写

* 对MyISAM表的读操作，不会阻塞其它进程对同一表的读请求，但会阻塞对同一表的写请求。只有当读锁释放后，才会执行其它进程的写操作。
* 对MyISAM表的写操作，会阻塞其它进程对同一表的读和写操作，只有当写锁释放后，才会执行其它进程的读写操作。

MyISAM不适合做写为主表的引擎，因为写锁后，其它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

### 3、行锁

会出现死锁，发生锁冲突几率低，并发高。

在MySQL的InnoDB引擎支持行锁，与Oracle不同，MySQL的行锁是通过索引加载的，也就是说，行锁是加在索引响应的行上的，要是对应的SQL语句没有走索引，则会全表扫描，行锁则无法实现，取而代之的是表锁，此时其它事务无法对当前表进行更新或插入操作。

CREATE TABLE `user` (

`name` VARCHAR(32) DEFAULT NULL,

`count` INT(11) DEFAULT NULL,

`id` INT(11) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`id`)

) ENGINE=INNODB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8

-- 这里，我们建一个user表，主键为id

-- A通过主键执行插入操作，但事务未提交

update user set count=10 where id=1;

-- B在此时也执行更新操作

update user set count=10 where id=2;

-- 由于是通过主键选中的，为行级锁，A和B操作的不是同一行，B执行的操作是可以执行的

-- A通过name执行插入操作，但事务未提交

update user set count=10 where name='xxx';

-- B在此时也执行更新操作

update user set count=10 where id=2;

-- 由于是通过非主键或索引选中的，升级为为表级锁，

-- B则无法对该表进行更新或插入操作，只有当A提交事务后，B才会成功执行

### 4、for update

如果在一条select语句后加上for update，则查询到的数据会被加上一条排它锁，其它事务可以读取，但不能进行更新和插入操作

-- A用户对id=1的记录进行加锁

select \* from user where id=1 for update;

-- B用户无法对该记录进行操作

update user set count=10 where id=1;

-- A用户commit以后则B用户可以对该记录进行操作

行锁的实现需要注意：

* 行锁必须有索引才能实现，否则会自动锁全表，那么就不是行锁了。
* 两个事务不能锁同一个索引。
* insert，delete，update在事务中都会自动默认加上排它锁。

行锁场景：

A用户消费，service层先查询该用户的账户余额，若余额足够，则进行后续的扣款操作；这种情况查询的时候应该对该记录进行加锁。

否则，B用户在A用户查询后消费前先一步将A用户账号上的钱转走，而此时A用户已经进行了用户余额是否足够的判断，则可能会出现余额已经不足但却扣款成功的情况。

为了避免此情况，需要在A用户操作该记录的时候进行for update加锁

### 5、扩展：间隙锁

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内并不存在的记录，叫做间隙

InnoDB也会对这个"间隙"加锁，这种锁机制就是所谓的间隙锁

-- 用户A

update user set count=8 where id>2 and id<6

-- 用户B

update user set count=10 where id=5;

如果用户A在进行了上述操作后，事务还未提交，则B无法对2~6之间的记录进行更新或插入记录，会阻塞，当A将事务提交后，B的更新操作会执行。

### 6、建议：

* 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
* 合理设计索引，尽量缩小锁的范围
* 尽可能减少索引条件，避免间隙锁
* 尽量控制事务大小，减少锁定资源量和时间长度