**一，MySQL的锁**

**数据库的锁——乐观锁+悲观锁**

**乐观锁**——用户自己实现，用户在执行操作时，很乐观，认为数据在一般情况不会冲突，

读数据的时候，认为别人不会修改数据，直接拿，**不上锁**

只有对数据提交更新的时候，才会对数据是否发生冲突进行检测，如果冲突了

，返回用户错误信息，让用户决定

——实现方法：版本号 ，时间戳

**悲观锁**——数据库实现，持悲观态度，认为数据一定会发生冲突，所以会给数据上锁

**数据库提供的锁机制——悲观锁**

**接下来讨论的都是悲观锁**

**二，MySQL的悲观锁机制**

**MySQL提供的悲观锁——表锁，页锁，行锁**

**MySQL Server提供表锁**

**页锁和行锁由存储引擎提供**

**MyISAM存储引擎——表锁**

——锁的粒度大，不会出现死锁，开销小，并发能力差

**InnoDB存储器引擎——行锁+表锁**

——行锁的粒度小，行锁可能会出现死锁，开销大，但是并发能力强

**——通过索引使用数据：行锁 不通过索引使用数据：表锁**

**三，MyISAM提供的悲观锁**

**表锁——表共享读锁 +表独占写锁**

**一个线程，读取某张表时——给这张表加上表共享读锁**

**一个线程，写某张表时——给这张表加上表独占写锁**

**一张表加上表共享读锁——不会阻塞其他线程的读操作，会阻塞其他线程的写操作**

**一张表加上表独占写锁——只有持有该锁的线程可以访问这张表，阻塞其他线程的所有操作**

**默认情况下 独占写锁优先级>共享都锁，什么意思？**

**假设 线程A正在更新文件，给表加了上独占写锁，这时候线程B来了，准备读取表，手里拿这个共享写锁，线程C也来了，准备更新表，手里拿着独占写锁，B和C一起等待A释**

**放锁，A释放锁后——C会先给表加上独占写锁（因为写锁优先级高于读锁）**

可以设置改变读锁和写锁的优先级：

* 通过指定启动参数low-priority-updates，使MyISAM引擎默认给予读请求以优先的权利。
* 通过执行命令SET LOW\_PRIORITY\_UPDATES=1，使该连接发出的更新请求优先级降低。
* 通过指定INSERT、UPDATE、DELETE语句的LOW\_PRIORITY属性，降低该语句的优先级。
* 给系统参数max\_write\_lock\_count设置一个合适的值，当一个表的读锁达到这个值后，MySQL就暂时将写请求的优先级降低，给读进程一定获得锁的机会。

**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**

**四，InnoDB提供的悲观锁**

InnoDB中的操作以事务为单位，默认情况下，一条SQL语句就是一个事务

在InnoDB表中，事务如果通过索引操作数据——行锁

事务如果不通过索引操作数据——表锁

**1，表锁**

InnoDB提供的表锁与MyISAM相同

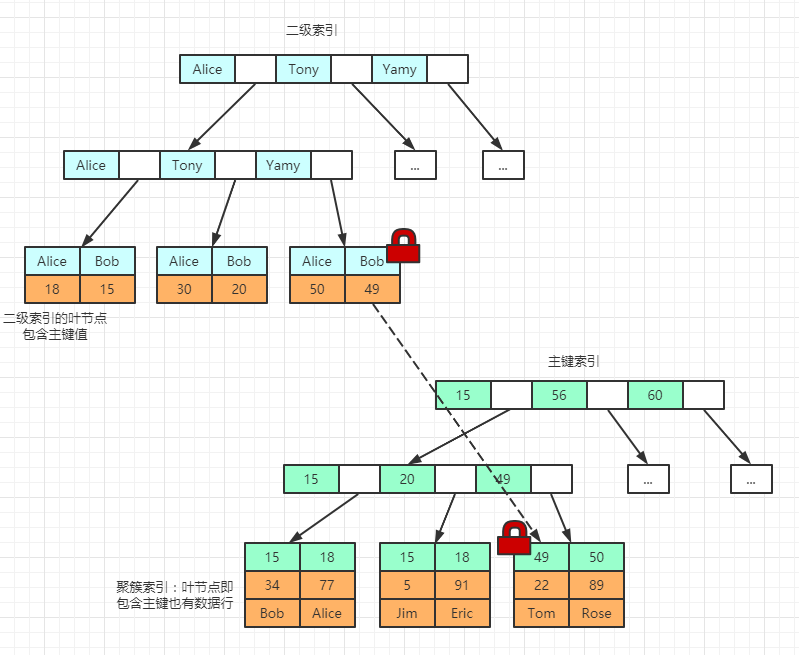
**2，行锁**

**行锁——是作用在索引上的——使用索引操作数据时，用行锁**

**如果通过主键索引访问某行记录，给主键索引表中的该行记录加上锁（1把）**

**如果通过辅助索引访问某行记录，首先给复制索引表中的对应的行记录加上锁**

**然后再给主键索引表中的目标记录加上所（2把）**



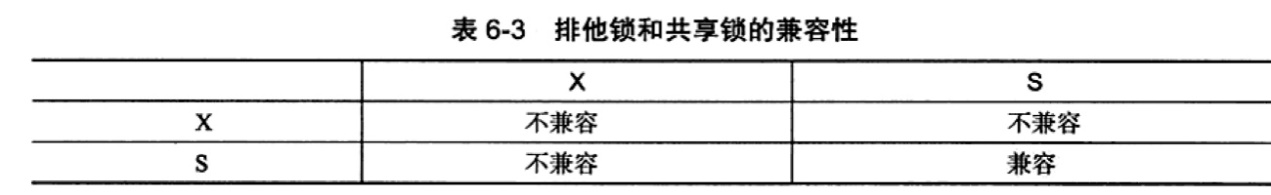
**InnoDB提供行锁——排他锁(X)+共享锁(S)**

**事务通过索引读取某张表中的记录——会给该记录加上共享锁**

**事务通过索引修改某张表中的记录——会给该记录加上排他锁**

**加上锁前，会检查该记录上是否存在锁——不存在，直接加上**

**——存在，判断存在的锁跟自己要加的锁是否兼容，兼容就加上，不兼容就等待锁的释放**



**3，意向锁（表级）**

**因为InnoDB同时存在表锁和行锁，可能出现一个事务通过索引访问表A中的某个记录，加上了行锁，同时，另一个事务没通过索引访问表A中的记录，准备给表A加表锁，这就出现了冲突**

**意向锁——解决行锁和表锁之间的冲突**

**意向锁是表记锁，只会阻塞表级的X和S，不会阻塞行级的X和S**

**有了意向锁后，行锁，表锁，意向锁的原理**

**如果一个事务通过索引要读一张表，会给这样表加表级的IS，在给记录加上行级的S**

**如果一个事务通过索引要修改一张表，会给这样表加表级的IX，在给记录加上行级的X**

**①在给表加上表级的IS/IX前，会检查表上是否存在表级锁，——不存在就直接加**

**——存在，判断存在的锁跟自己要加的IS/IX是否兼容，兼容直接加，不兼容就阻塞**

**②在给记录加上行级的X或T前，会检查记录上是否存在行级锁——不存在直接加**

**——存在，就判断存在的锁跟自己要加的锁是否兼容，兼容直接加，不兼容就阻塞**



**如果一个事务没有通过索引要读一张表，会给这样表加表级的S**

**如果一个事务没有通过索引要修改一张表，会给这样表加表级的X**

**再给表加上表记的X或S前，会检查该表上是否存在表级锁——不存在，直接加**

**——存在，就判断存在的锁跟自己要加的锁是否兼容，兼容就加上，不兼容就阻塞**

**InnoDB 存储引擎的锁的算法有三种：**

* Record lock：单个行记录上的锁
* Gap lock：间隙锁，锁定一个范围，不包括记录本身
* Next-key lock：record+gap 锁定一个范围，包含记录本身