**然后是关于数据库的各种锁的总结：**

1.共享锁（又称读锁）、排它锁（又称写锁）：

InnoDB引擎的锁机制：**InnoDB支持事务，支持行锁和表锁用的比较多，Myisam不支持事务，只支持表锁。**

共享锁（S）：允许一个事务去读一行，阻止其他事务获得相同数据集的排他锁。  
排他锁（X)：允许获得排他锁的事务更新数据，阻止其他事务取得相同数据集的共享读锁和排他写锁。  
意向共享锁（IS）：事务打算给数据行加行共享锁，事务在给一个数据行加共享锁前必须先取得该表的IS锁。  
意向排他锁（IX）：事务打算给数据行加行排他锁，事务在给一个数据行加排他锁前必须先取得该表的IX锁。

说明：

**1）共享锁和排他锁都是行锁，意向锁都是表锁，应用中我们只会使用到共享锁和排他锁，意向锁是mysql内部使用的，不需要用户干预。**

**2）对于UPDATE、DELETE和INSERT语句，InnoDB会自动给涉及数据集加排他锁（X)；对于普通SELECT语句，InnoDB不会加任何锁，事务可以通过以下语句显示给记录集加共享锁或排他锁。**  
共享锁（S）：SELECT \* FROM table\_name WHERE ... LOCK IN SHARE MODE。  
排他锁（X)：SELECT \* FROM table\_name WHERE ... FOR UPDATE。

\*\*对于锁定行记录后需要进行更新操作的应用，应该使用Select...For update 方式，获取排它锁。（用共享锁，在读了之后再写会阻塞，会导致死锁）

**这里说说Myisam：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。**

**3）InnoDB行锁是通过给索引上的索引项加锁来实现的，因此InnoDB这种行锁实现特点意味着：只有通过索引条件检索数据，InnoDB才使用行级锁，否则，InnoDB将使用表锁！**

2.乐观锁、悲观锁：

**悲观锁：**悲观锁，正如其名，它指的是对数据被外界（包括本系统当前的其他事务，以及来自外部系统的事务处理）修改持保守态度，因此，在整个数据处理过程中，将数据处于锁定状态。悲观锁的实现，往往依靠数据库提供的锁机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统中实现了加锁机制，也无法保证外部系统不会修改数据）

1）使用悲观锁，我们必须关闭mysql数据库的自动提交属性，采用手动提交事务的方式，因为MySQL默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。

2）需要注意的是，在事务中，只有SELECT ... FOR UPDATE 或LOCK IN SHARE MODE 同一笔数据时会等待其它事务结束后才执行，一般SELECT ... 则不受此影响。对于UPDATE、DELETE和INSERT语句，InnoDB会自动给涉及数据集加排他锁（X)。

3）补充：MySQL select…for update的Row Lock与Table Lock

使用select…for update会把数据给锁住，不过我们需要注意一些锁的级别，MySQL InnoDB默认Row-Level Lock，所以只有「明确」地指定主键（或有索引的地方），MySQL 才会执行Row lock (只锁住被选取的数据) ，否则MySQL 将会执行Table Lock (将整个数据表单给锁住)。

**乐观锁：**

乐观锁（ Optimistic Locking ） 相对悲观锁而言，乐观锁假设认为数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则让返回用户错误的信息，让用户决定如何去做（**一般是回滚事务**）。那么我们如何实现乐观锁呢，一般来说有以下2种方式：

1）.使用数据版本（Version）记录机制实现，这是乐观锁最常用的一种实现方式。何谓数据版本？即为数据增加一个版本标识，一般是通过为数据库表增加一个数字类型的 “version” 字段来实现。当读取数据时，将version字段的值一同读出，数据每更新一次，对此version值加一。当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的version值进行比对，如果数据库表当前版本号与第一次取出来的version值相等，则予以更新，否则认为是过期数据。

2）.乐观锁定的第二种实现方式和第一种差不多，同样是在需要乐观锁控制的table中增加一个字段，名称无所谓，字段类型使用时间戳（timestamp）, 和上面的version类似，也是在更新提交的时候检查当前数据库中数据的时间戳和自己更新前取到的时间戳进行对比，如果一致则OK，否则就是版本冲突。

**总结：两种锁各有优缺点，不可认为一种好于另一种，像乐观锁适用于写比较少的情况下，即冲突真的很少发生的时候，这样可以省去了锁的开销，加大了系统的整个吞吐量。但如果经常产生冲突，上层应用会不断的进行retry，这样反倒是降低了性能，所以这种情况下用悲观锁就比较合适。**

**另外，高并发情况下个人认为乐观锁要好于悲观锁，因为悲观锁的机制使得各个线程等待时间过长，极其影响效率，乐观锁可以在一定程度上提高并发度。**

3.表锁、行锁

表级锁(table-level locking)：MyISAM和MEMORY存储引擎

行级锁(row-level locking) ：InnoDB存储引擎

页面锁(page-level-locking)：BDB存储引擎

**表级锁：**开销小，加锁快;不会出现死锁;锁定粒度大，发生锁冲突的概率最高,并发度最低。

**行级锁：**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。

**页面锁：**开销和加锁时间界于表锁和行锁之间;会出现死锁;锁定粒度界于表锁和行锁之间，并发度一般。

之后看到了其他的知识，会继续补充。