# 分布式锁

针对分布式锁的实现，目前比较常用的方案：

1. 基于数据库实现分布式锁
2. 基于缓存（redis、memcache、tair）实现分布式锁
3. 基于Zookeeper实现分布式锁

# MySQL数据库锁

InnoDB存储引擎既支持行级锁（row-level locking），也支持表级锁，但默认情况下采用行级锁。

表级锁：开销小，加锁快，不会产生死锁，锁粒度大，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，锁粒度小，发生锁冲突的概率最低，并发度最高。

页面锁：开销和加锁时间介于表锁和行锁之间，会产生死锁，锁粒度介于表锁和行锁之间，并发度一般。

综上所述，很难笼统地说哪种锁更好，只能就具体应用的特点选择合适的锁！仅从锁的角度来看，表级锁更适合查询为主，只有少量按索引更新数据的应用场景，如web应用；而行级锁则更适合于有大量按照索引条件并发更新少量不同数据，同时又有并发查询的应用，如一些在线事务处理（OLAP）系统。

## 表锁

MySQL的表级锁有两种模式：表共享读锁（Table Read Lock）和表独占写锁（Table Write Lock）。

使用lock table/unlock table进行加锁/解锁。

## 行锁

InnoDB实现了两种类型的行锁：

共享锁（S）：又称读锁。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。若事务T对数据对象A加上S锁，则事务T可以读A但不能修改A，其他事务只能再对A加S锁，而不能加X锁，直到T释放A上的S锁。这保证了其他事务可以读A，但在T释放A上的S锁之前不能对A做任何修改。

所谓的共享锁，就是多个事务只能读数据不能修改数据。加共享锁可以使用select…lock in share mode语句。

排他锁（X）：又称写锁。允许获取排他锁的事务更新数据，阻止其他事务获取相同的数据集共享读锁和排他写锁。若事务T对数据对象A加上X锁，事务T可以读A也可以修改A，其他事务不能再对A加任何锁，知道T释放A上的锁。

排他锁指的是一个事务在一行数据加上排他锁后，其他事务不能再在其上加上其他的锁。InnoDB引擎默认的修改数据语句：update、delete、insert都会自动给涉及到的数据加上排他锁。所以加过排他锁的数据行在其他事务中是不能修改数据的，也不能通过for update和lock in share mode锁的方式查询数据，但是可以直接通过select…from…查询数据，因为普通查询没有任何锁限制。

另外，为了允许行锁和表锁共存，实现多粒度的锁机制，InnoDB还有两种内部使用的意向锁（Intention Locks），这两种意向锁都是表锁。

意向共享锁（IS）：事务打算给数据行共享锁，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。