MySQL锁

全局锁

全局锁就是对整个数据库实例加锁

MySQL 提供了一个加全局读锁的方法，命令是 Flush tables with read lock (FTWRL)

全局锁的典型使用场景是，做全库逻辑备份，在备份过程中整个库完全处于只读状态。

官方自带的逻辑备份工具是 mysqldump。当 mysqldump 使用参数–single-transaction 的时候，导数据之前就会启动一个事务，来确保拿到一致性视图。而由于 MVCC 的支持，这个过程中数据是可以正常更新的。

有了这个功能，为什么还需要 FTWRL 呢?

一致性读是好，但前提是引擎要支持这个隔离级别。

single-transaction 方法只适用于所有的表使用事务引擎的库

既然要全库只读，为什么不使用 set global readonly=true 的方式呢?

一是，在有些系统中，readonly 的值会被用来做其他逻辑，比如用来判断一个库是主库还是备库。因此，修改 global 变量的方式影响面更大，我不建议你使用。

二是，在异常处理机制上有差异。如果执行 FTWRL 命令之后由于客户端发生异常断开，那么 MySQL 会自动释放这个全局锁，整个库回到可以正常更新的状态。而将整个库设置为 readonly 之后，如果客户端发生异常，则数据库就会一直保持 readonly 状态，这样会导致整个库长时间处于不可写状态，风险较高。

表锁

表锁

表锁的语法是 lock tables … read/write。

与 FTWRL 类似，可以用 unlock tables 主动释放锁，也可以在客户端断开的时候自动释放。需要注意，lock tables 语法除了会限制别的线程的读写外，也限定了本线程接下来的操作对象。

元数据锁（meta data lock，MDL)

MDL 不需要显式使用，在访问一个表的时候会被自动加上

MDL 的作用是，保证读写的正确性

当对一个表做增删改查操作的时候，加 MDL 读锁；当要对表做结构变更操作的时候，加 MDL 写锁。

读锁之间不互斥，因此你可以有多个线程同时对一张表增删改查。

读写锁之间、写锁之间是互斥的，用来保证变更表结构操作的安全性。因此，如果有两个线程要同时给一个表加字段，其中一个要等另一个执行完才能开始执行。

事务不提交，就会一直占着 MDL 锁

行锁

MySQL 的行锁是在引擎层由各个引擎自己实现的。但并不是所有的引擎都支持行锁，比如 MyISAM 引擎就不支持行锁

InnoDB 是支持行锁的，这也是 MyISAM 被 InnoDB 替代的重要原因之一

两阶段锁协议。

在 InnoDB 事务中，行锁是在需要的时候才加上的，但并不是不需要了就立刻释放，而是要等到事务结束时才释放。

如果你的事务中需要锁多个行，要把最可能造成锁冲突、最可能影响并发度的锁的申请时机尽量往后放。

死锁

当并发系统中不同线程出现循环资源依赖，涉及的线程都在等待别的线程释放资源时，就会导致这几个线程都进入无限等待的状态

当出现死锁以后，有两种策略：

一种策略是，直接进入等待，直到超时。这个超时时间可以通过参数 innodb_lock_wait_timeout 来设置。

在 InnoDB 中，innodb_lock_wait_timeout 的默认值是 50s，意味着如果采用第一个策略，当出现死锁以后，第一个被锁住的线程要过 50s 才会超时退出，然后其他线程才有可能继续执行。

另一种策略主动死锁检测是，发起死锁检测，发现死锁后，主动回滚死锁链条中的某一个事务，让其他事务得以继续执行。将参数 innodb_deadlock_detect 设置为 on，表示开启这个逻辑。

死锁检测存在的问题：消耗大量的CPU资源

一种头痛医头的方法，就是如果你能确保这个业务一定不会出现死锁，可以临时把死锁检测关掉。

另一个思路是控制并发度

减少死锁的主要方向，就是控制访问相同资源的并发事务量。