1.mysql的锁介绍

数据库锁设计的初衷是处理并发问题，在出现并发访问的时候数据库需要合理的控制资源的访问规则，锁就是用来实现这些访问规则的重要数据结构。根据加锁的范围，mysql的锁可以分为全局锁、表锁和行锁三类。

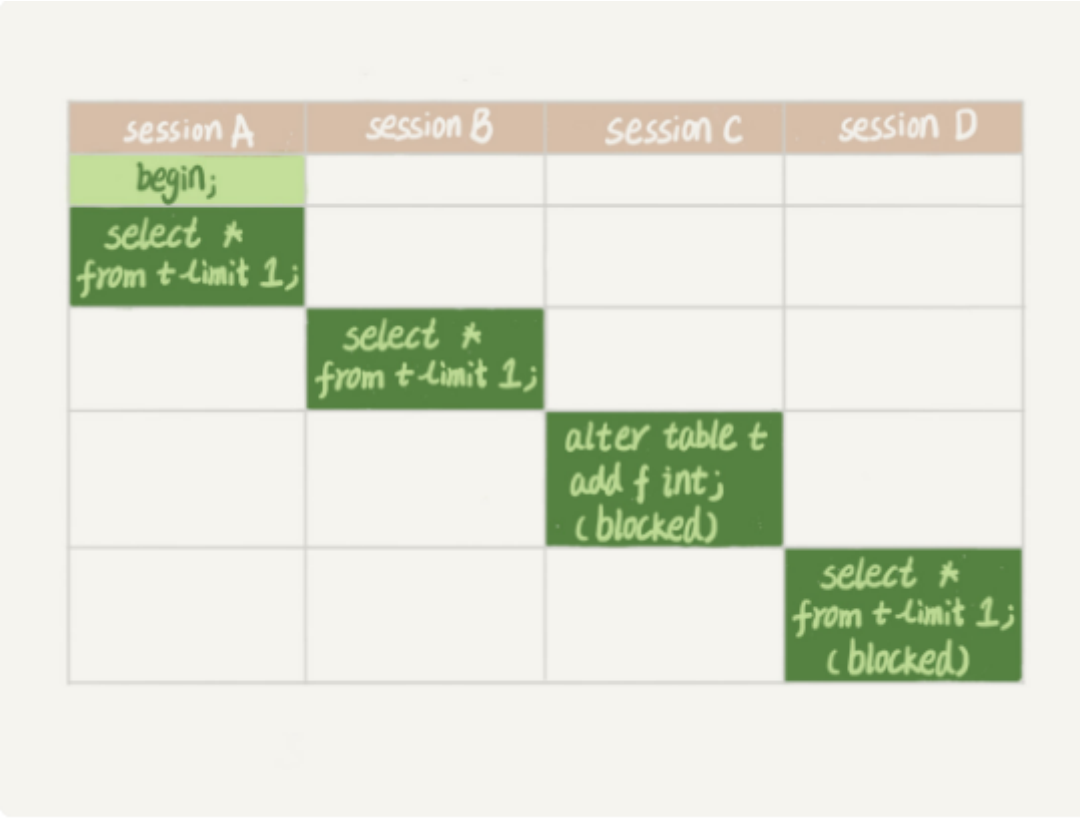
1.1 全局锁

全局锁是对整个数据库实例加锁。使用**Flush tables with read lock(FTWRL)** 命令就可以给整个数据库实例加上全局锁，其他线程的增删改、建表、修改表结构、更新类事务的提交语句都会被阻塞。全局锁的典型使用场景就是做全库逻辑备份。如果在主库做备份，会导致业务停摆。如果在从库做备份，备库不能执行从主库传来的binlog，会导致主从延迟。如果不加锁做备份会导致备份期间的数据不一致，后续用备份恢复数据时会造成数据逻辑错误。 如果全库都是使用事务引擎的数据表，可以使用官方自带的备份工具mysqldump使用参数 **–single-transaction** 在可重复读级别下，利用数据库MVCC的特性，导数据会启动一个事务，拿到一致性视图，其他线程也可正常更新数据。

1.2 表级锁

mysql里面的表级锁有两种：表锁和元数据锁(meta data lock,MDL). 表锁的用法是 **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。对于InnoDB这种支持行锁的引擎来说，一般不建议使用表锁。 另一类表级锁是MDL。在mysql5.5版本之后引入了MDL锁，访问表的时候会自动加上，保证读写的正确性。当对表增删改查的时候，自动加上MDL读锁；对表结构作修改的时候，自动加上MDL写锁。

- 读锁之间不互斥，因此可以多个线程对表做增删改查。
- 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。



如图所示，sessionA取得了MDL读锁，读锁之间不互斥，因为sessionB可以顺利执行，sessionC是对表结构的操作，加上了MDL写锁，互斥，因次sessionC被阻塞，sessionD读锁与sessionC的写锁互斥被阻塞。事务中的MDL锁，在语句开始时执行，语句结束并不会马上释放，而是等到事务结束之后释放。 所以一旦数据库中存在长事务，这个时候要做DDL（修改数据库表结构的操作）可能会拖垮整个库。建议做DDL操作前先查询是否存在长事务，如果存在先等待或者kill掉长事务。但是假如对一张热点表做DDL操作，可能kill掉长事务并不能解决实际问题，这个时候可以在sql里设定语句等待时间，在指定的等待时间内拿到写锁就执行，拿不到也不会阻塞后面的业务语句执行。sql如下：

```
ALTER TABLE tbl_name NOWAIT add column ...
ALTER TABLE tbl_name WAIT N add column ...
```

1.3 行锁

mysql的行锁是由各个引擎实现的，但不是所有引擎都支持行锁，myISAM就不支持。并发控制不支持行锁的情况，任何时刻表上只能有一个更新操作，这样效率太低了。InnoDB是支持行锁的引擎，这也是myISAM被替代的原因之一。

1.3.1 两阶段锁

在InnoDB的事务中，行锁并不是从事务开始时加上的，而是在需要时(更新操作语句执行前)加上的，但是锁释放是在事务提交后释放的，这就是InnoDB的两阶段锁协议。所以在一个事务中，如果要锁多个行，要把最可能造成锁冲突，影响并发度的锁放后放，这样他们在锁等待期间对业务并发度造成的影响最小。

1.3.2 死锁和死锁检测

当并发系统中不同线程都出现了循环资源依赖，涉及的线程都在等其他线程释放资源，导致线程进入了无限等待，这就叫死锁。

如图所示，事务A持有id=1的行锁，等待事务B持有的id=2的行锁释放，但是事务B又在等待事务A持有的id=1的行锁释放，进入了死锁状态。 有两种策略处理死锁：

- 设置超时时间，等待超时后直接回滚释放锁。这个可以通过参数**innodb_lock_wait_timeout**设置
- 发起死锁检测，发现死锁后，主动回滚死锁链条其中一个事务。将参数**innodb_deadlock_detect**设置为on。

在InnoDB中，innodb_lock_wait_timeout的默认值为50s，对于在线服务来说，这个时间不可接受。但又不能设置的太短，如果设置为1s，有可能这只是锁等待的正常时间，会出现误伤。这个需要对业务做压测之后设置一个合理值。 正常情况下还是采用第二种策略，进行死锁检测。死锁检测也会带来额外的开销，假如有1000个并发事务对同一行操作，那么死锁检测操作就是100万。死锁检测在并发冲突多时会大量的消耗cpu资源，cpu利用率很高，但是每秒却执行不了几个事务。 解决由热点行更新导致的性能问题有两个思路：

- 如果能确保业务不会出现死锁，可以临时关掉死锁检测。但是这种操作会带来一定风险，毕竟关掉死锁检测之后如果出现死锁会出现大量超时，这是业务有损的。
- 控制并发度。如果能在mysql端控制好并发的线程数量，就不会造成死锁检测消耗大量的计算资源。(使用中间件或修改mysql源码等实现)