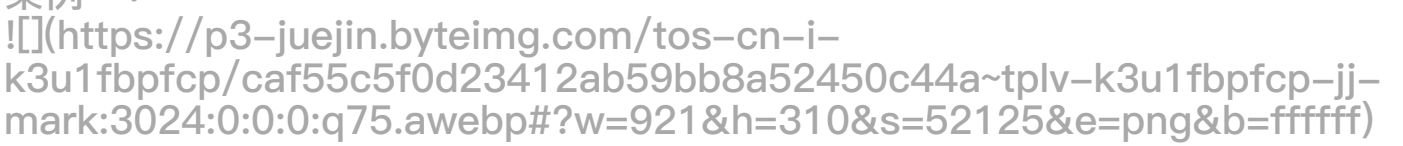
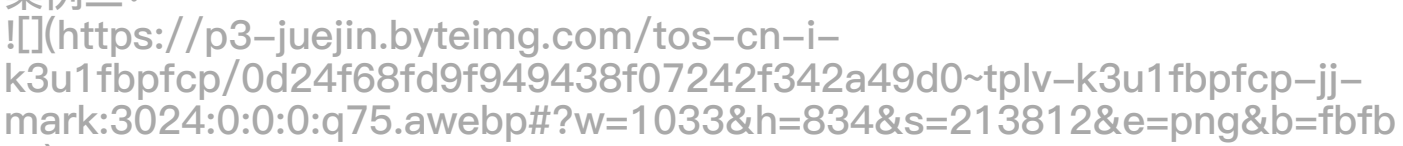
【面试题】细说mysql中的各种锁

前言

作为一名IT从业人员，无论你是开发，测试还是运维，在面试的过程中，我们经常会被数据库，数据库中最经常被问到就是MySql。当面试官问MySql的时候经常会问道一个问题，”MySQL中有哪些锁？“当我们去网上找相应的资料的时候却是五花八门，张三说的一个样，李四说的另一个样。
like this....

案例一：


案例二：


案例三：

案例四：

搞得我们都不知道他们说到底对不对，全不全，.....于是我们非常之困惑，面试

的时候不知如何作答，正是为了解决这个问题，我查阅了很多书籍，国内外资料，写下这篇文章，希望对你有所帮助。

锁到底是个什么东西

维基百科是这样解释的：锁是一种保安设施，是人类为了保护自己的财产而发明的一种用钥匙才能开启的装置。就现代而言，是一种以钥匙、密码、电路或者其他用具来开启的封缄装置，用以防止物品被打开、移走兼具防护、管理甚至是装饰的作用。所以锁最重要的作用自然就是保护安全。

在计算机的世界里其实也一样，使用锁的目的几乎都是为了保证数据的安全可靠。在生活中锁千奇百怪，但我们可以将锁大致分为两种，一种是需要钥匙的，一种是不需要钥匙的，当然现代家庭中的指纹锁，数字密码锁都是需要钥匙的哈，只是钥匙不是实体钥匙而已。有很多朋友可能很纳闷，锁还有不需要钥匙的？答案是：有的
比如：

如果你来自农村，并且家里还有老宅那你可能也见过这样的锁：

这些锁都是不需要钥匙的，我们将之为”闩（shuan）锁“。
需要钥匙的锁就很常见了，比如：

再比如：

这种需要钥匙的锁我将之称为”锁“。

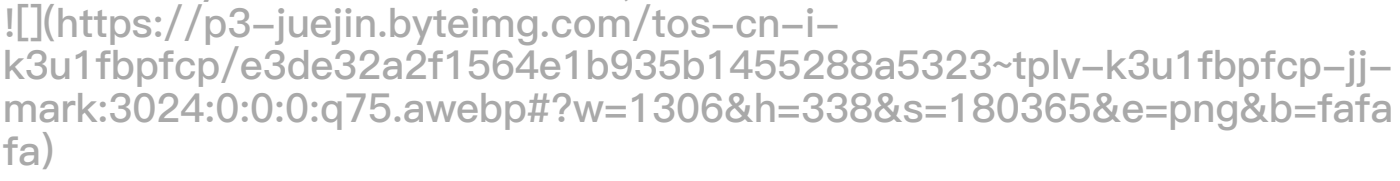
所以非常直观的感受就是虽然都为锁，但是闩锁比较”轻量“，自然也就没那么安全，而”锁“则麻烦一些，需要随身带着钥匙，所以相对也会更安全一些。相信到这里你已经对锁有比较好的理解了，那我们接着说数据库中的锁。

Mysql中的锁

在数据库中自然也将锁分成了两类，**一类叫latch也就是闩锁，另一种叫lock（锁）。**

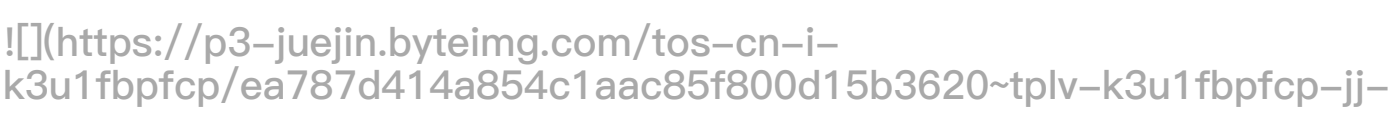
latch一般称为闩锁（轻量级的锁），因为其要求锁定的时间必须非常短。若持续的时间长，则应用的性能会非常差。在InnoDB存储引擎中，latch又可以分为mutex（互斥量）和rwlock（读写锁）。其目的是用来保证并发线程操作临界资源的正确性，并且通常没有死锁检测的机制。
在数据库中,连接是一种珍贵的资源，连接自然不可能无限变多，也不可能同时被多个线程同时持有，这时候就需要用锁来保证同一时刻同一个连接只能被一个线程持有，这时候用到的锁就是我们说的闩锁。

而**lock**的对象是事务，用来锁定的是数据库中的对象，如表、页、行。并且一般lock的对象仅在事务commit或rollback后进行释放（不同事务隔离级别释放的时间可能不同）。此外，lock，正如在大多数数据库中一样，是有死锁机制的。所以我们在看一些文章的时候经常会看到将MySQL的锁分为行锁、表锁、页锁，他们得出的依据就是来来自于此。（关于行锁、表锁、页锁我们先按下不表，后面展开说！）

这里借用MySQL技术内幕中一张图，对比一下latch与lock的区别：


总结一下：其实他俩最大的区别就是作用的范围不同，latch更像是我们日常开发过程中使用到的应用程序级别的“普通锁”。而lock则是为了操作数据库中数据的一种特殊的“事务锁”，平时我们面试的时候面试官问的也主要是这种为了操作数据库中数据而存在的事务锁。

在这里先提前给大家总结一下：



mark:3024:0:0:0:q75.awebp#?w=994&h=199&s=25456&e=png&b=f3f3f5
)

站在更高的维度看问题-锁无非就是读和写

不知你是否发现，无论是生活中的各种鸟锁，**无非就是要不要用到钥匙，然后把他分成闩锁和锁**。到了计算机界，锁无非也总的来说也就是两种，一种是我拿到了锁我（一个事务）只让你（另一个事务）读只让你读，不让你写，另一种是：哥们我拿到锁了你们都往后靠一靠吧（连读都不让你读的），等我操作完了你们再来吧，好比厕所蹲坑，不可一个坑两个人同时蹲。所以你要明白“有空一起拉屎”只是调侃，可千万不能当真！

![](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/897553f3459f4ace858f44d9f4da45ec~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=172&h=180&s=40612&e=png&b=f6f5f4
)

回归正题，于是乎按照严格我拿到锁了，你到底能不能读这个权限，将锁分为了两种：

* 共享锁（S Lock）：也叫做读锁
* 排他锁（X Lock），也叫做写锁

注：你必须明白一点写的权限高于读，所以如果我说基于读这个权限，意思就是读都读不了，还写个毛啊！

所以我们总结再总结一下可以得出：

![](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/0d4af177c69b4028a229a9dfa6e23b24~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=637&h=210&s=16069&e=png&b=f3f3f5
)

等等...我听说的全局锁、表锁、行锁你丫的没给我讲呢

![](https://p3-juejin.byteimg.com/tos-cn-i-

k3u1fbpfcp/92276a93215e4b20bef93097ac5753c5~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=690&h=690&s=49277&e=jpg&b=5fbbe3)
前面我们已经讲了，锁无非就是读和写，这是从功能上来说，但是每个事务都有自己的表现形式呀，你说对吧?

在MySQL中，数据库中的数据被分为了”库-->表--->行”，所以MySQL中的锁，按照锁的粒度分自然而然就产生对应的全局锁、表锁和行锁了哟!

* 全局锁：锁定数据库中的所有表
* 表级锁：每次操作锁住整张表
* 行级锁：每次操作锁住对应的行数据

好了有了上面的铺垫，我们继续讲解这三种锁的使用，以便你在工作和面试中“惊艳四座、大杀四方、godlike！”

全局锁就是对整个数据库实例加锁，加锁后整个实例就处于只读状态，后续的DML的写语句，DDL语句，已经更新操作的事务提交语句都将被阻塞。

“后续的DML的写语句，DDL语句，已经更新操作的事务提交语句都将被阻塞。”这几句话列位能理解的吧?

好吧，你们都能理解，那我也当个小丑讲一下吧：

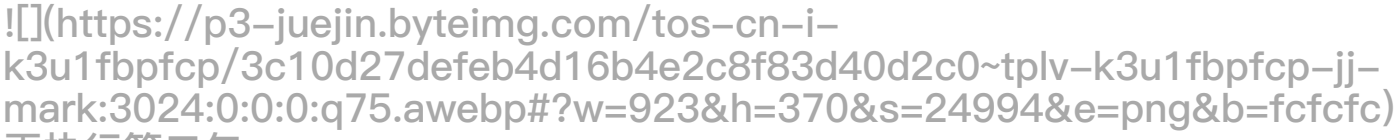
DML的写语句：sql语句中的update和delete语句都是写语句

DDL语句：就是要修改表结构的语句，比如`ALTER TABLE table_name ADD column_name data_type`
已经更新操作的事务提交语句：这一句优点难理解,直接上代码：

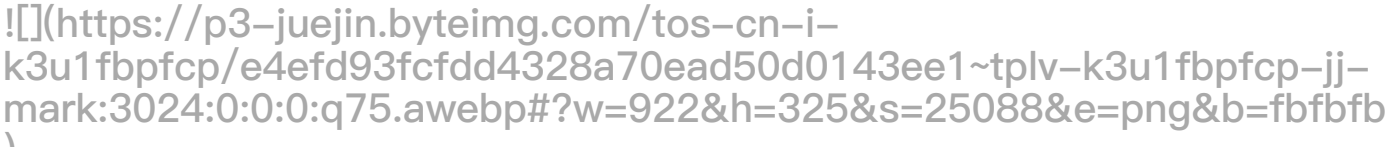
```
BEGIN -- 第一句

UPDATE `subject` SET `name`='ccc12' WHERE id =1 -- 第二句

COMMIT;--第三句
```

他想表达的意思就是，在锁库之前已经执行了前面两句sql然后库被锁住了，这时候如果再执行第三句，那么这个操作就会被阻塞！！

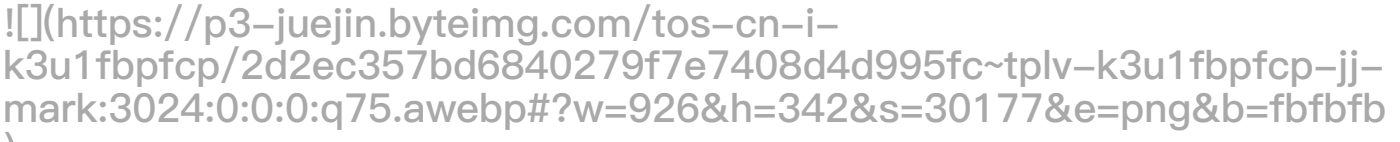
证明：
首先我们先执行第一句：



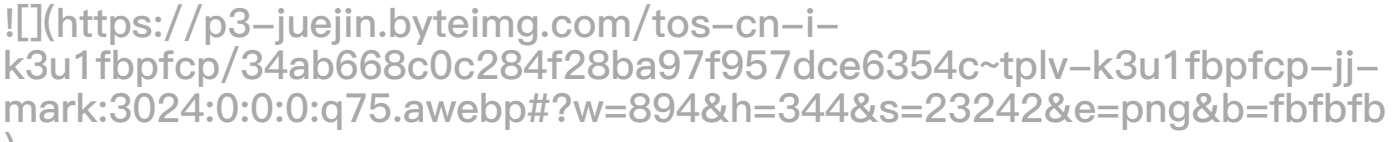
再执行第二句：



好的，我们去另一个事务中把库锁了：

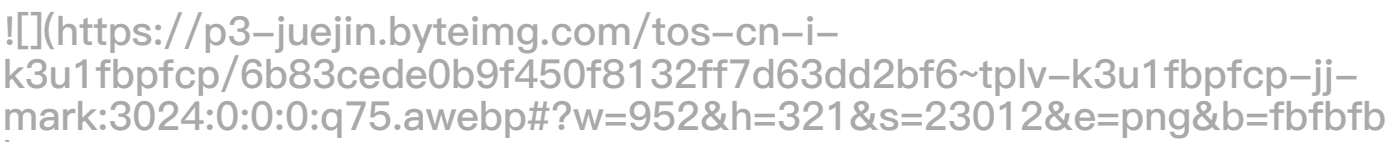


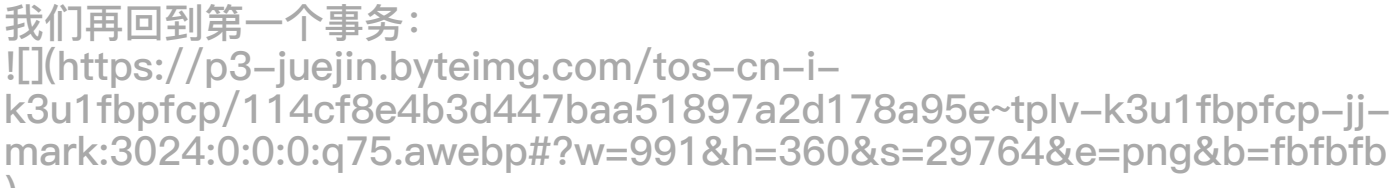
这时候我们再去第一个事务中执行提交语句：



哎呀，家人们呐，看到了没，他被阻塞住了，不执行了！

我们再把锁释放了：



我们再回到第一个事务：


COMMIT已经执行成功了，从花费时间上也不难看出确实是被阻塞了呢！

全局锁这么猛(锁的粒度太™的大了)，他在什么地方使用呢?

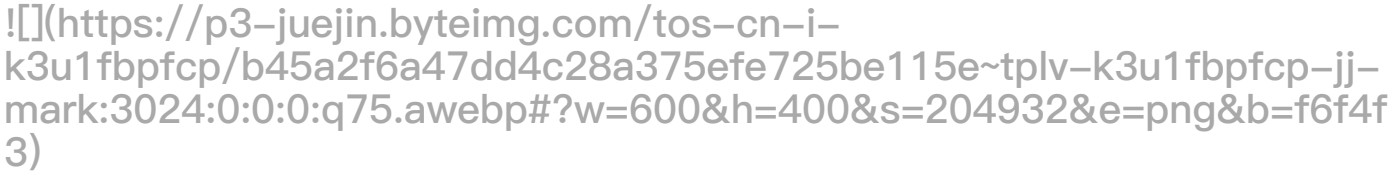
全局锁典型的使用场景是做全库的逻辑备份，对所有的表进行锁定，从而获取一致性视图，保证数据的完整性。但是在数据库中加全局并不是一个好选择，因为数据库中加全局锁，是一个比较重的操作，存在以下问题：

* 1．如果在主库上备份，那么在备份期间都不能执行更新，业务基本上就得停摆。
* 2.如果在从库上备份，那么在备份期间从库不能执行主库同步过来的二进制日志（binlog），会导致主从延迟。

在InnoDB引擎中，我们可以在备份时加上参数--single-transaction参数来完成不加锁的一致性数据备份。

`mysqldump --single-transaction -uroot -p123456 test>test.sql`

说完全局锁，我们接着说**表锁**，对，你没看错，是表锁，不是表嫂！



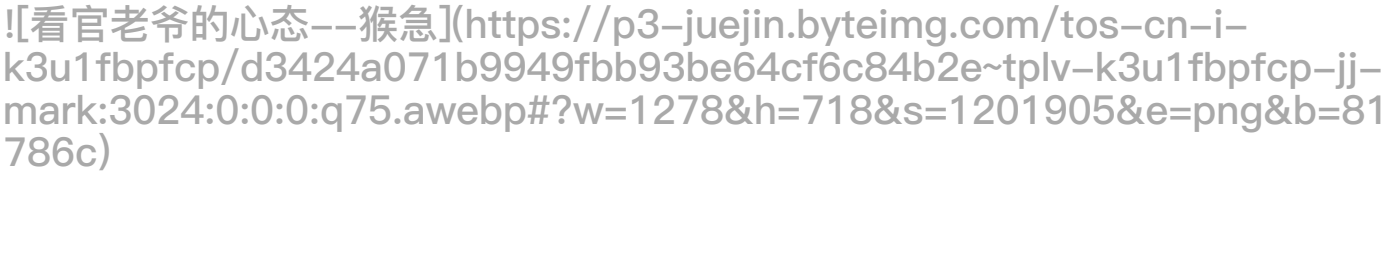
但是**在这里我们要把它称为表级锁，而不是表锁**，为什么要这么称呼呢？那肯定有是有道理滴，因为
同为表级锁，主要分为以下三类：

* 1.表锁
* 2.元数据锁（meta datalock，MDL）

* 3.意向锁

他们都可以称为表（级）锁，**对于表级锁来说，每次操作锁住整张表，在MyISAM、InnoDB、BDB等存储引擎中都有实现**。由于每次操作锁住整张表，所以表锁的锁定粒度大，发生锁冲突的概率最高，并发度低。

这下真的要开始讲表锁了！！



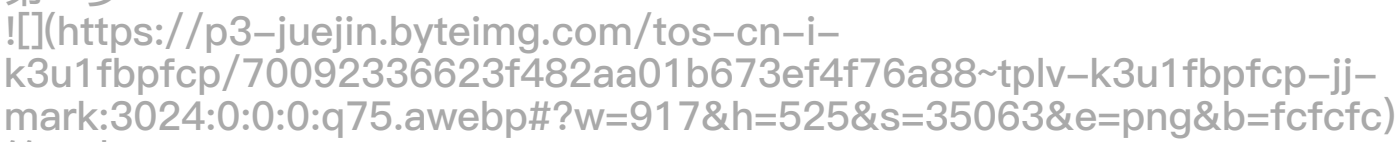
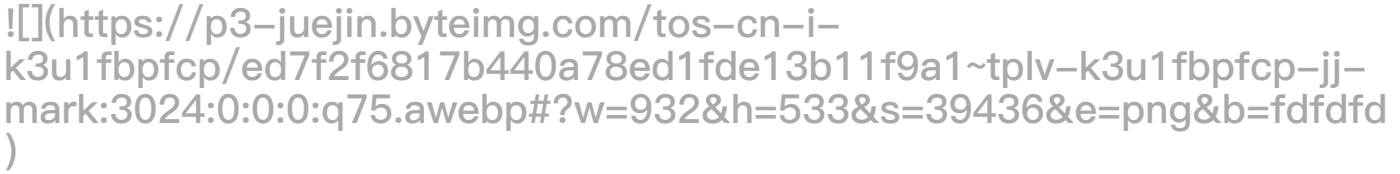
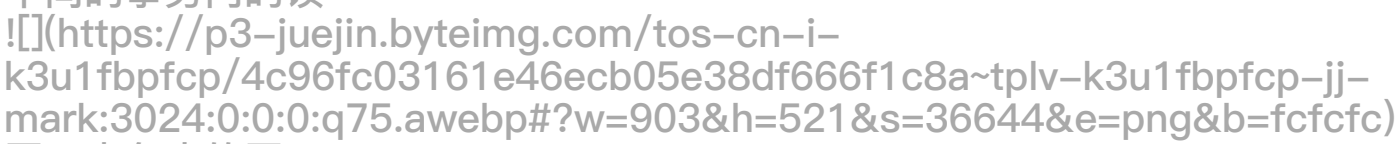
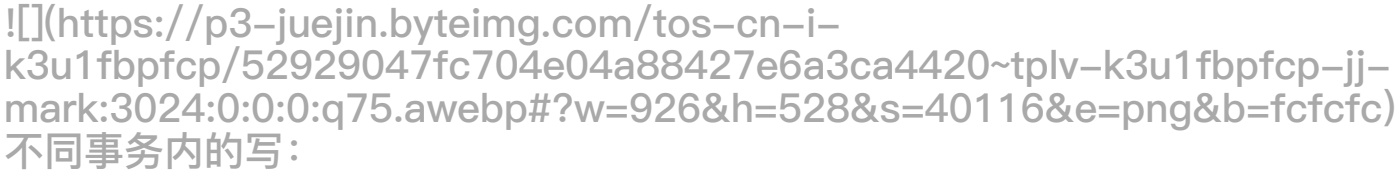
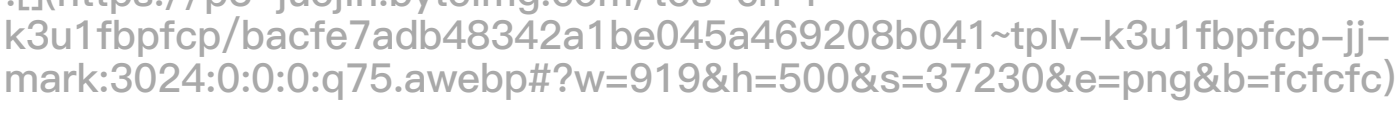
表锁主要分为两类：

* 1.表共享读锁（read lock）
* 2.表独占写锁（write lock）

共享读锁：当一张表被上了共享读锁以后，这张表只能被读不能被写，且读的话大家（不同的事务）都可以读，写的话大家都不能写。
举个例子好理解一点：

我们先将表上锁：

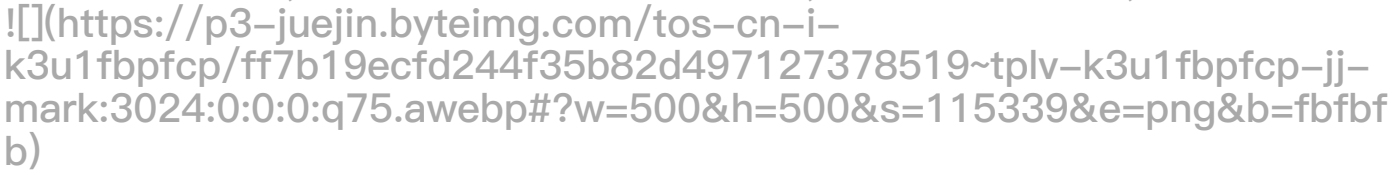
```
BEGIN
LOCK TABLES `subject` READ

SELECT * FROM `subject`  WHERE id=1

COMMIT;
```

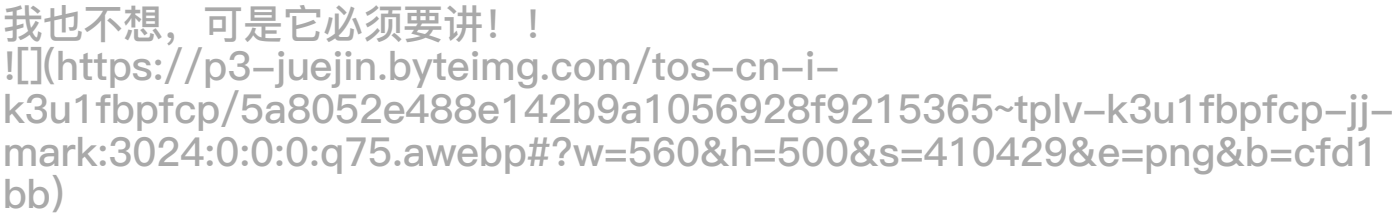
同一个事务内的读，如图：
第一步：

第二步：

不同的事务内的读：

同一事务内的写：

不同事务内的写：


独占写锁：当一张表被上了共享读锁以后，当前事务既能读数据也能写数据，同时还能修改表结构，别的事务既不能读数据也不能写数据，必须等到该锁被释放以后才能读写数据。（验证方式和上面一样，不再给图了）

这里一定要注意：**我锁强调的是”别的事务“，而不是有些文章或者教学视频说的别的客户端！这是完全不同的概念，同一个客端可以上开启多个不同的事务！**

总结如图：

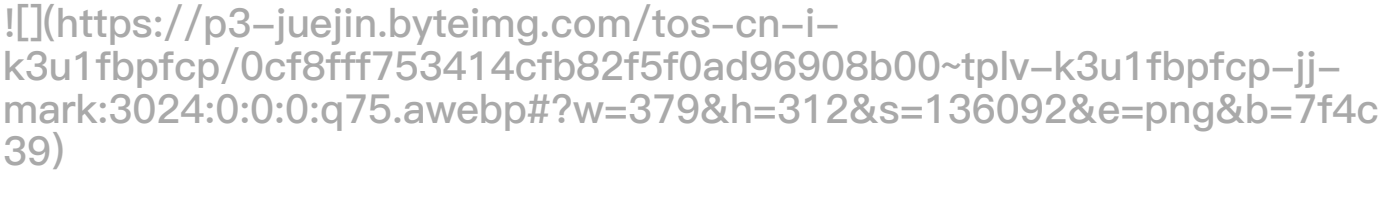

表锁终于说完了，有些人看到这这里内容太多，估计已经暴躁了，想骂人！！


我也不想，可是它必须要讲！！


我们接着讲另一个表级锁--元数据锁

元数据锁（metadata lock） 也叫MDL，他的加锁过程是系统自动控制，无需显式使用，在访问一张表的时候会自动加上。MDL锁主要作用是维护表元数据的数据一致性，在表上有活动事务的时候，不可以对元数据进行写入操作。为了避免DML与DDL冲突，保证读写的正确性。在MySQL5.5中引I入了MDL，当对一张表进行增删改查的时候，加MDL读锁（共享)；当对表结构进行变更操作的时候，加MDL写锁(排他)。

说白了，元数据锁其实就是”表结构锁“没人这么叫，你可以这么理解，意思就是数据增删改查的时候能动了表结构，不然就会产生冲突，比如你在插入数据的时候，你把我表结构的其中一列给删了，你这让我这么插？



好的，我们继续讲解最后一个表级锁--**意向锁**

意向锁,为了避免DML在执行时，加的行锁与表锁的冲突，在InnoDB中引入了意向锁，使得表锁不用检查每行数据是否加锁，使用意向锁来减少表锁的检查。

考虑这个例子：事务A锁住了表中的一行，让这一行只能读，不能写。之后，事务B申请整个表的写锁。如果事务B申请成功，那么理论上它就能修改表中的任意一行，这与A持有的行锁是冲突的。数据库需要避免这种冲突，就是说要让B的申请被阻塞，直到A释放了行锁。

数据库要怎么判断这个冲突呢?

step1：判断表是否已被其他事务用表锁锁表

step2：判断表中的每一行是否已被行锁锁住。

注意: step2这样的判断方法效率实在不高，因为需要遍历整个表。于是就有了意向锁。在意向锁存在的情况下，事务A必须先申请表的意向共享锁，成功后再申请一行的行锁。
在意向锁存在的情况下，上面的判断可以改成step1：不变,step2：发现表上有意向共享锁，说明表中有些行被共享行锁锁住了，因此，事务B申请表的写锁会被阻塞。

意向锁分为两类:

* 1.意向共享锁（IS）：由语句select...lock in sharemode添加。与表锁共享锁（read）兼容，与表锁排它锁（write）互斥。
* 2.意向排他锁（Ix）：由insert、update、delete、select...forupdate添加。与表锁共享锁（read）及排它锁（write）都互斥,意向锁之间不会互斥。

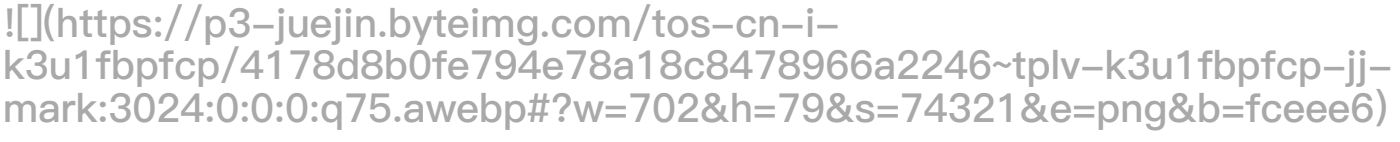
可以通过以下SQL，查看意向锁及行锁的加锁情况：select object_schema,object_name,index_name,lock_type,lock_mode,lock_data from performance_schema.data_locks;

终于到了”行级锁“——千呼万唤始出来

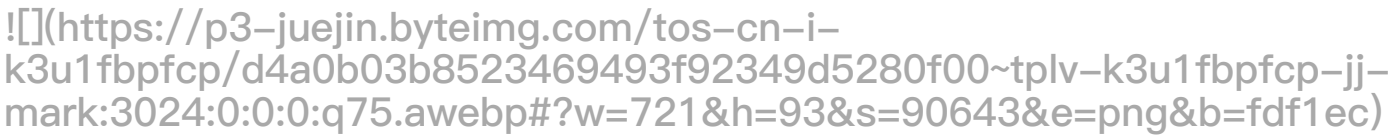
我们继续学习**行级锁**，行级锁每次操作锁住对应行的数据，**锁定粒度最小，发生锁冲突的概率最低，并发度最高。** InnoDB存储引擎支持行级锁，InnoDB的数据是基于索引组织的，**行锁是通过对索引上的索引项加锁来实现的，而不是对记录加的锁。**（这一点非常重要）

对于行级锁，主要分为以下三类：

* **1.行锁（RecordLock）：**锁定单个行记录的锁，防止其他事务对此行进行update和delete。在RC（读已提交）、RR（可重复读）隔离级别下都支持。



* **2.间隙锁（GapLock）：**锁定索引记录间隙（不含该记录），确保索引记录间隙不变，防止其他事务在这个间隙进行insert，产生幻读。在RR隔离级别下都支持。



* **3.临键锁（Next-KeyLock）：**行锁和间隙锁组合，同时锁住数据，并锁住数据前面的间隙Gap。在RR隔离级别下支持。

行锁也被叫做记录锁，InnoDB实现了以下两种类型的行锁：

1.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它锁。

2.排他锁（X）： 允许获取排他锁的事务更新数据，阻止其他事务获得相同数据集的共享锁和排他锁。

其中insert、update、delete语句MySQL会自动为这些语句自动加上排他锁，select...lock in share mode语句会加上共享锁，select...for update加上排他锁，普通的select语句并不会加任何锁！

锁的升级与降级

默认情况下，InnoDB在REPEATABLE READ事务隔离级别运行，InnoDB使用next-key锁进行搜索和索引扫描，以防止幻读。

第一类情况：

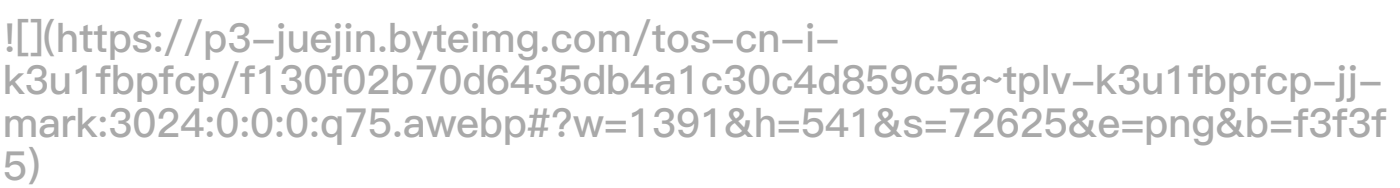
* 针对唯一索引进行检索时，对已存在的记录进行等值匹配时，将会自动优化为行锁。
* InnoDB的行锁是针对于索引加的锁，不通过索引条件检索数据，那么InnoDB将对表中的所有记录加锁，此时就会升级为表锁。

第二类情况：

* 索引上的等值查询(唯一索引)，给不存在的记录加锁时，优化为间隙锁。
* 索引上的等值查询(普通索引)，向右遍历时最后一个值不满足查询需求时，next-key lock退化为间隙锁。
* 索引上的范围查询(唯一索引)，会访问到不满足条件的第一个值为止。

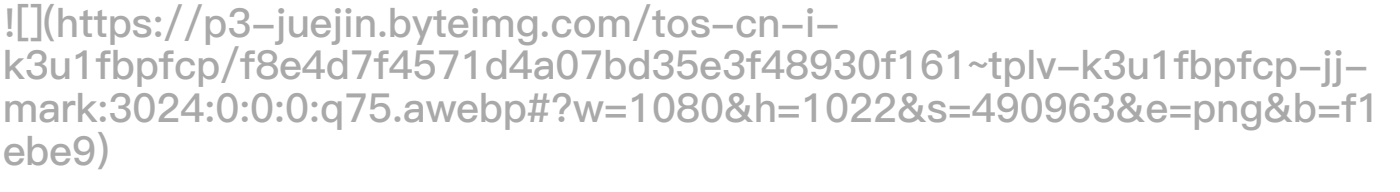
注意： *间隙锁唯一目的是防止其他事务插入间隙。间隙锁可以共存，一个事务采用的间隙锁不会阻止另一个事务在同一间隙上采用间隙锁。*

所以最后总结一下就是这样的：



最后
--

如果你能看到这里，那说明你真是太棒了，全文快五千字了，基本上把mysql的锁讲透了，同时你还有什么问题可以给我留言，一起进步！能一键三连的话，我将不甚感激！



原文链接: https://juejin.cn/post/7356809873432215615