> 点赞再看，养成习惯，微信搜索【**三太子敖丙**】关注这个互联网苟且偷生的工具人。
>
> 本文 **GitHub** https://github.com/JavaFamily 已收录，有一线大厂面试完整考点、资料以及我的系列文章。

前言

数据库存在几种`事务隔离级别`我想不用我说，大家也都知道的吧?

什么？还不知道？还不知道就自己去补课，我默认大家都知道了。算了我是暖男，在贴一下给大家看看，下次可别忘了哈。

有四种：

- `读未提交（READ UNCOMMITTED）`：一个事务还没提交时，它做的变更就能被别的事务看到。
- `读提交（READ COMMITTED）`：一个事务提交之后，它做的变更才会被其他事务看到。
- `可重复读（REPEATABLE READ）`：一个事务执行过程中看到的数据，总是跟这个事务在启动时看到的数据是一致的。当然在可重复读隔离级别下，未提交变更对其他事务也是不可见的。
- `串行化（SERIALIZABLE）`：对于同一行记录，“写”会加“写锁”，“读”会加“读锁”，当出现读写锁冲突的时候，后访问的事务必须等前一个事务执行完成，才能继续执行。

隔离级别解决了哪些问题大家也应该都是知道的分别有：

- `脏读（dirty read）`：如果一个事务读到了另一个未提交事务修改过的数据。

事务A	事务B
事务开始	总数100
	事务开始
	插入新数据
查询表A所有数据总数101	
提交事务	
	回滚

- 不可重复读（non-repeatable read）：如果一个事务只能读到另一个已经提交的事务修改过的数据，并且其他事务每对该数据进行一次修改并提交后，该事务都能查询得到最新值。

事务A	事务B
查询表A所有数据总数100	
	插入新数据
查询表A所有数据总数101	
	插入新数据
查询表A所有数据总数102	

- 幻读（phantom read）：如果一个事务先根据某些条件查询出一些记录，之后另一个事务又向表中插入了符合这些条件的记录，原先的事务再次按照该条件查询时，能把另一个事务插入的记录也读出来。

如何设置事务的隔离级别?

我们可以通过下边的语句修改事务的隔离级别:

```
SET [GLOBAL|SESSION] TRANSACTION ISOLATION LEVEL level;//等级就是上面的几种
```

怎么启动的?

启动事务一般就是:

1. 显式启动事务语句， begin 或 start transaction，配套的提交语句是commit，回滚语句是rollback。
2. set autocommit=0，这个命令会将这个线程的自动提交关掉，意味着如果你只执行一个select语句，这个事务就启动了，而且并不会自动提交。这个事务持续存在直到你主动执行commit 或 rollback 语句，或者断开连接。

我在书中看到都是不要建议大家使用自动的，我看了一下我们的场景，倒是很少主动启动事务的，因为我们单个库的场景不多，更多都是分布式事物。

但是 `长事务` 是大家需要注意的，因为一旦set autocommit=0自动开启事务，所有的查询也都会在事务里面了，有慢SQL那数据库也容易被拖垮的。

我最近就遇到了这样的问题，数据库经常接到报警，其中就有长事务导致的问题。

视图

首先得说一下，我后面所有的知识都是基于 `InnoDB` 的，因为MyISAM不支持事务。

`视图` ，这是事务隔离实现的根本，数据库里面会创建一个视图，访问的时候以视图的逻辑结果为准。

在MySQL里，有两个“视图”的概念:

- 一个是view，它是一个用查询语句定义的虚拟表，在调用的时候执行查询语句并生成结果。创建视

图的语法是create view … ，而它的查询方法与表一样。
- 另一个是InnoDB在实现MVCC时用到的一致性读视图，即consistent read view，用于支持RC（Read Committed，读提交）和RR（Repeatable Read，可重复读）隔离级别的实现。

在 `可重复读` 隔离级别下，这个视图是在事务启动时创建的，整个事务存在期间都用这个视图。

在正式介绍之前，我还需要介绍一下 `版本链` 。

版本连

在 `可重复读` 隔离级别下，事务在启动的时候就“拍了个快照”，注意，这个快照是基于整库的。

你肯定会说，这怎么可能？如果一个库有100G，那么我启动一个事务，MySQL就要拷贝100G的数据出来，这个过程得多慢啊，这谁顶得住啊？可是你回头一想，平时的事务执行起来不是很快么？

实际上，数据库并不需要拷贝出这100G的数据，那快照怎么实现的？

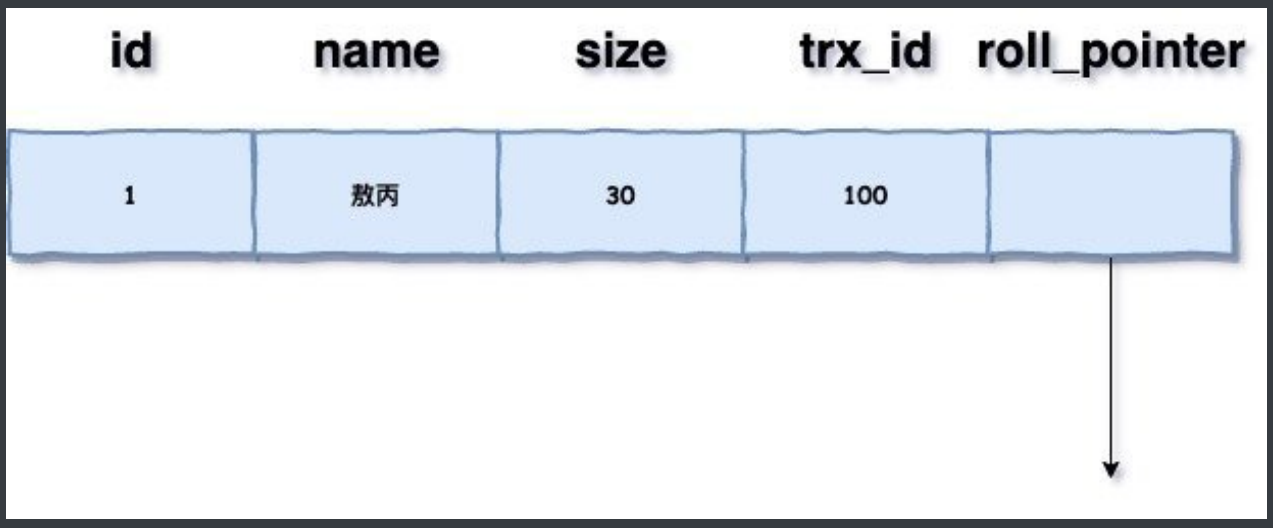
InnoDB里面每个事务有一个唯一的事务ID，叫作 `transaction id` ，它是在事务开始的时候向InnoDB的事务系统申请的，是按申请顺序严格递增的。

每行数据也都是有多个版本的，每次事务更新数据的时候，都会生成一个新的数据版本，并且把 `transaction id` 赋值给这个数据版本的事务ID，记为 `row trx_id` 。同时，旧的数据版本要保留，并且在新的数据版本中，能够有信息可以直接拿到它。

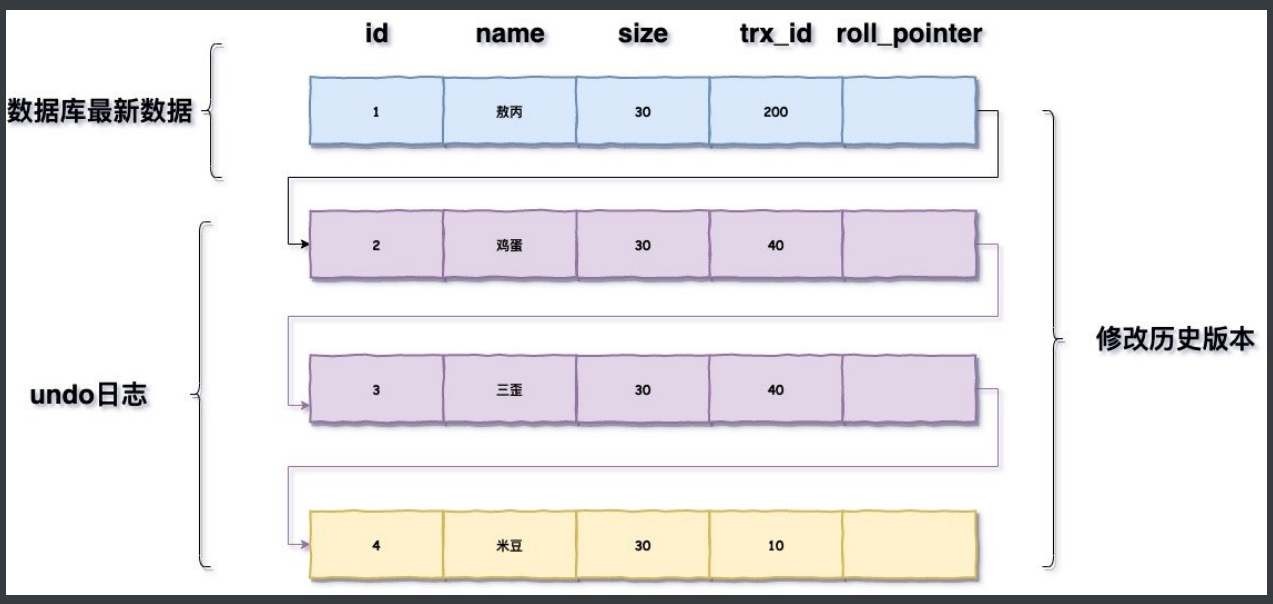
也就是说，数据表中的一行记录，其实可能有多个版本(row)，每个版本有自己的 `row trx_id` 。

这是一个隐藏列，还有另外一个 `roll_pointer` ： 每次对某条聚簇索引记录进行改动时，都会把旧的版本写入到 `undo日志` 中，然后这个隐藏列就相当于一个指针，可以通过它来找到该记录修改前的信息

两者都在InnoDB的聚簇索引中，大概就长这样：

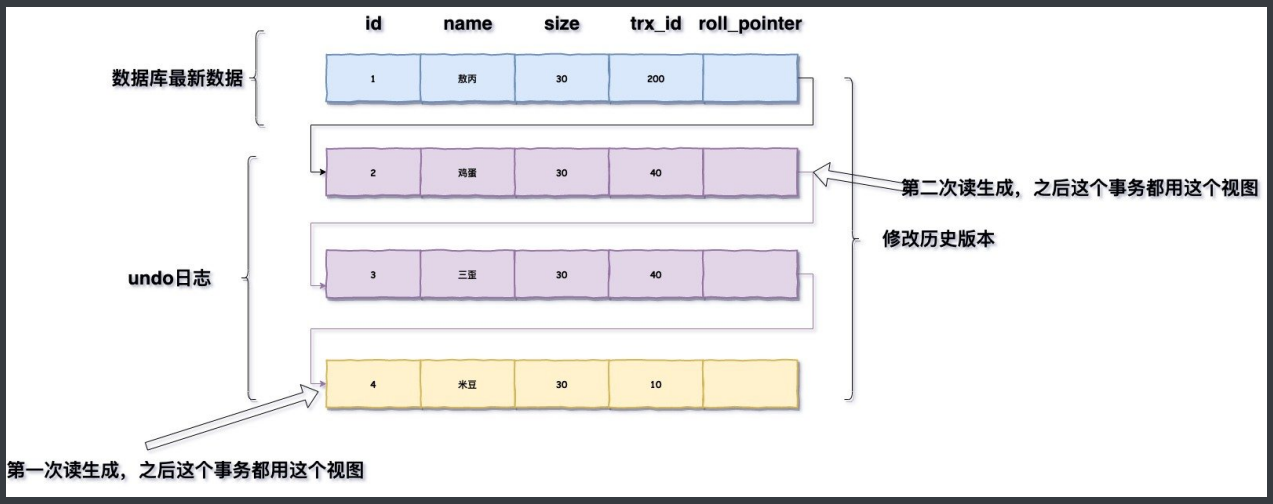


`undo log` 的回滚机制也是依靠这个版本链，每次对记录进行改动，都会记录一条undo日志，每条undo日志也都有一个 `roll_pointer` 属性（INSERT操作对应的undo日志没有该属性，因为该记录并没有更早的版本），可以将这些undo日志都连起来，串成一个链表，所以现在的情况就像下图一样：

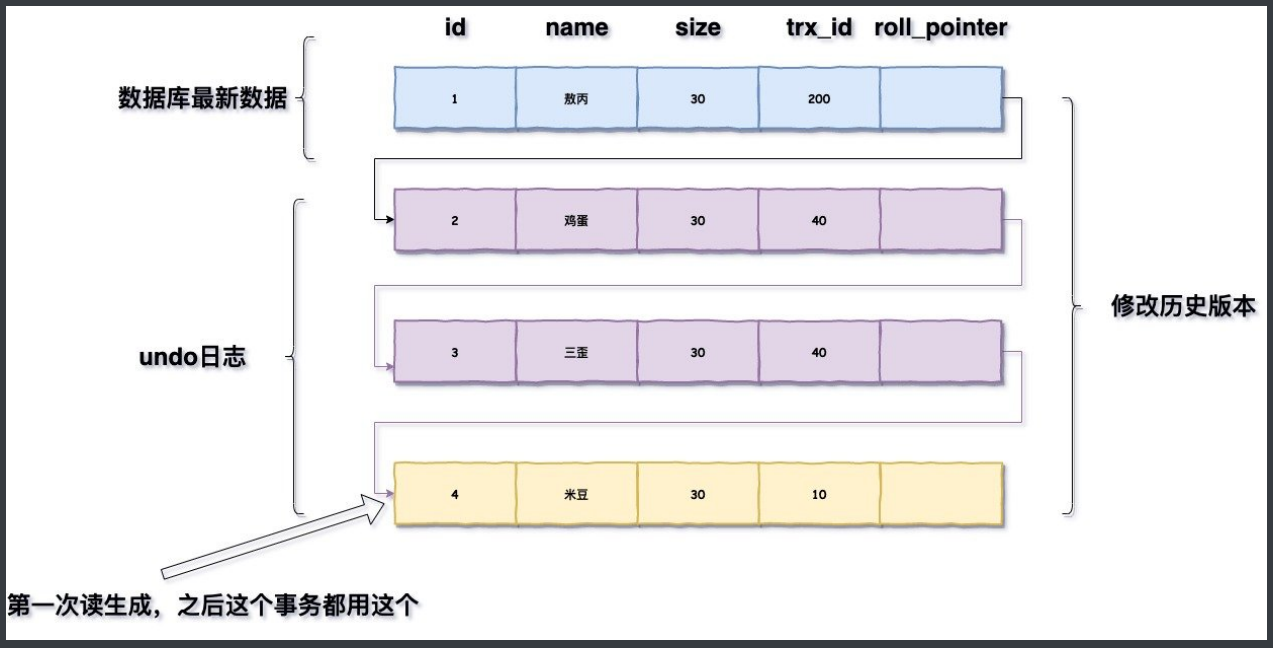


接下来可以说一下事务隔离级别和MVCC的关系了，下面的例子是一个版本链，事务id大家可以看出，三个事务分别作了不同的事情。

在 读提交 隔离级别下，这个视图是在每个SQL语句开始执行的时候创建的，在这个隔离级别下，事务在每次查询开始时都会生成一个独立的ReadView。



可重复读，在第一次读取数据时生成一个ReadView，对于使用 REPEATABLE READ 隔离级别的事务来说，只会在第一次执行查询语句时生成一个 ReadView ，之后的查询就不会重复生成了，所以一个事务的查询结果每次都是一样的。



这里需要注意的是， 读未提交 隔离级别下直接返回记录上的最新值，没有视图概念，也就是图中丙丙那一栏，脏读，幻读，不可重复读都有可能发生。

而 串行化 隔离级别下直接用加锁的方式来避免并行访问。

所以之前有小伙伴我问的时候经常说错，mvcc里面跟事务隔离级别相关的，只有可重复读和读已提交这两种。

我工作以来，我所有接触的公司的数据库隔离级别默认都是读已提交，不过我们会在一些场景开启可重复读，序列化很少见。

可重复读我们之前都是在跟订单金额相关的场景去开启的，还有很多数据修改过程也会用可重复度，因为很多值是需要查询出来，依据那个值做别的操作的，如果多次查询的结果值不一样，那后者也会受到影响。

`序列化` 被称为数据库隔离级别的 `黄金标准` ，它是绝大多数商业数据库系统中提供的最高隔离级别，一些高度广泛部署的系统甚至无法提供隔离级别与可序列化一样高，金融的场景居多，性能也是最差的，但是银行取钱你会在乎那几秒么？

有没有发现银行的ATM响应速度特别慢，他们的场景都是很严密的，各种事务，锁，都是结合的，就是为了保证结果的准确性。

大家可以用这个命令去看看自己公司或者自己现在使用的数据库的隔离级别：

```
show variables
```

资料参考：《MySQL 是怎样运行的：从根儿上理解 MySQL》、《高性能MySQL》、《 MySQL 实战 45 讲》

总结

从上边的描述中我们可以看出来，所谓的 `MVCC（Multi-Version Concurrency Control ，多版本并发控制）` 指的就是在使用 `读已提交（READ COMMITTD）、可重复读（REPEATABLE READ）` 这两种隔离级别的事务在执行普通的SELECT操作时访问记录的版本链的过程，这样子可以使不同事务的读-写、写-读操作并发执行，从而提升系统性能。

这两个隔离级别的一个很大不同就是： `生成ReadView的时机不同` ，READ COMMITTD在每一次进行普通SELECT操作前都会生成一个ReadView，而REPEATABLE READ只在第一次进行普通SELECT操作前生成一个ReadView，数据的可重复读其实就是ReadView的重复使用。

这样去解释这些技术，主要是希望大家对现象背后的本质多点思考，不然你去背出这几种隔离级别，以及各种数据现象是没有任何意义的，实际开发过程中真的出现了问题，你不懂本质以及过程，你去排查也会很难受的，到头来还是要看书，看资料。

我是敖丙，一个在互联网苟且偷生的程序员。

你知道的越多，你不知道的越多，人才们的 **【三连】** 就是丙丙创作的最大动力，我们下期见！

注：如果本篇博客有任何错误和建议，欢迎人才们留言！

> 文章持续更新，可以微信搜索「 **三太子敖丙** 」第一时间阅读，回复【**资料**】有我准备的一线大厂面试资料和简历模板，本文 **GitHub** https://github.com/JavaFamily 已经收录，有大厂面试完整考点，欢迎Star。