链滴

mysql MVCC 多版本并发控制

作者：xiaokedamowang
原文链接：https://ld246.com/article/1628676803640
来源网站：链滴

mysql MVCC多版本并发控制

在研究MVCC之前需要先了解2个概念

1. 当前读

- 查询操作

select * from xx **for update** (排他锁)或者 select * from xx **lock in share mode**(共享锁)

我们平时一般不用这个

- 增删改

增删改都是当前读操作, 修改语句都需要先读取数据, 判断约束是否正确, 比如主键索引 唯一索引

当前读总是读取到最新的数据

2. 快照读

普通的select * from xx 平时一般我们用这个

快照读读取的是历史版本数据

一. 复习 不可重复读 和 幻读

1. 不可重复读

1个事务**第一次**查询了某条记录,然后过了一会**再次**查询的时候发现同样ID的记录和第一次查询的不一样

解决方式 隔离级别调整成 **可重复读**

2. 幻读

1个事务**第一次**查询列表返回10条记录,然后过了一会**再次**查询的时候发现变成了11条记录

解决方式 隔离级别调整成 **串行化**

- **隔离级别 只针对于快照读**
- **注意红色字体, 产生不可重复读 和 幻读的前提 都是查询了2次**, 也就说说中间的时间点,其他的事干了1些别的事情

二. 隐藏列

我们创建的表,其实有2-3个隐藏的列,**隐藏主键**(如果表有自己的主键,就没有这列),**事务ID**,**回滚指针**

(用户表)

用户名	**密码**	**邮箱**	**年龄**	**隐藏主键** DB_ROW_ID	**事务ID** DB_TRX_ID	**回滚指针** DB_ROLL_PTR
张三	***	***	11	1	?	?
李四	***	***	12	2	?	?

三. readview

在进行快照读的时候,会生成一个readview读视图

readview有4个属性

**活跃的事务ID集合
(就是没有commit的事务)　　列表中的最小事务ID**
未分配的下一个事务ID　　创建该视图的事务ID

PS:为什么用中文 因为网上查到的字段有2种, 可能是因为数据版本问题导致的变量名不一样, 但是意思是一样的

四. undolog

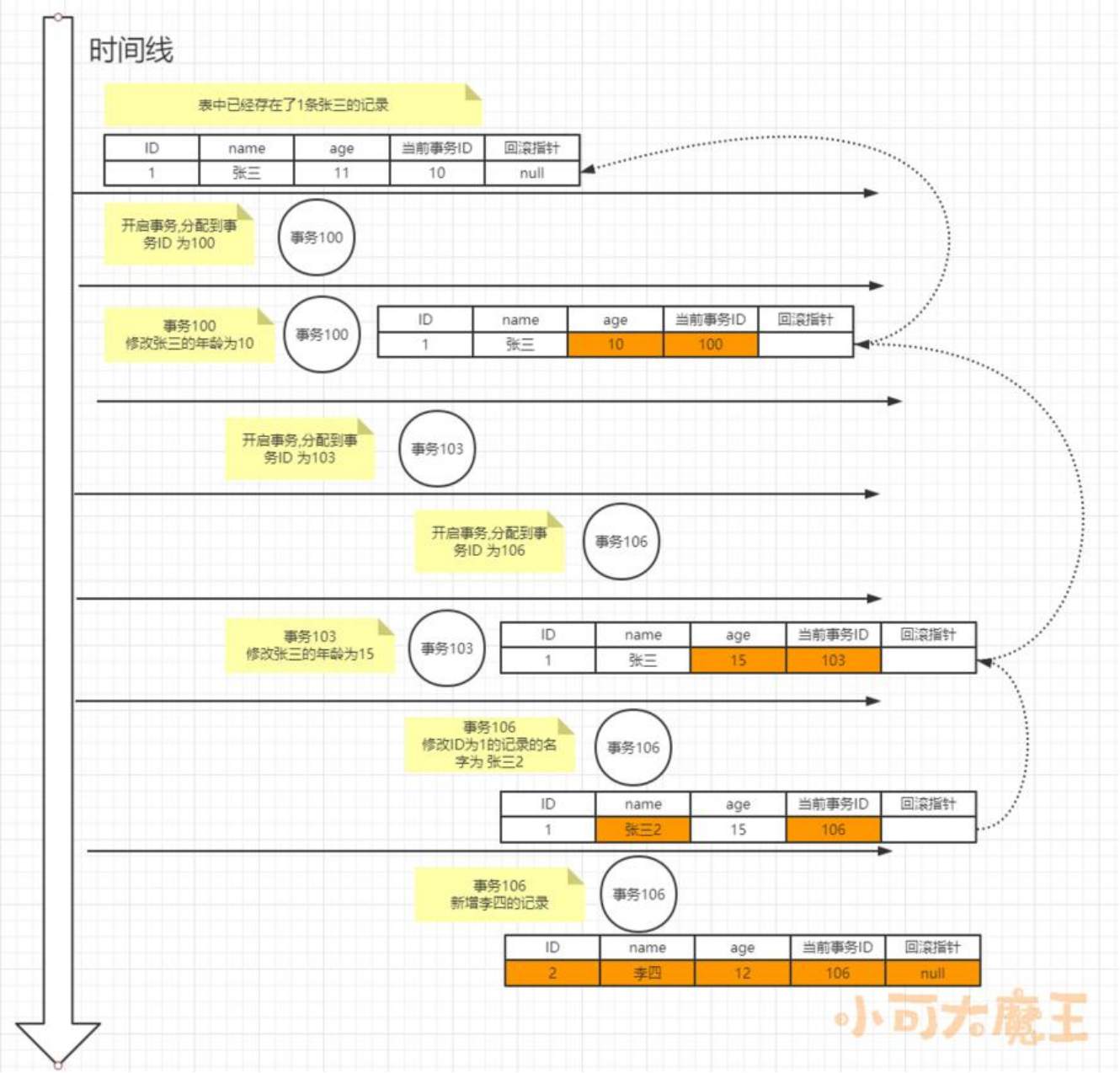
在事务中增删改, 都会生成undolog

每次修改的回滚指针都会指向上一个版本的记录,如果rollback,就会找到对应的记录恢复

undolog不会在commit之后立刻删除,而是放入待清理的链表,由purge线程判断是否有其他事务在使用来决定是否删除

如下图:

原文链接：mysql MVCC 多版本并发控制



四. 可见性算法(比较的规则)

1. 当前记录的 事务ID是否小于readview列表中最小事务ID,如果小于,代表了这条记录在这个readview创建之前就已经commit了,所以能看见否则进行下一步判断

2. 当前记录的 事务ID是否**大于等于**readview的尚未分配的下一个事务ID,如是大于等于就代表这条记录是在readview之后生成的,所以不能看到,否则进入一下一步判断

3. 当前记录的 事务ID是否在活跃的事务ID集合中, 如果在,就说明这条记录在readview创建的时候还没有commit,也就是说当前事务不能看到记录,否则就能看到

五. 案例分析1

在下面时间线之前已经存在了一条张三的记录

原文链接：mysql MVCC 多版本并发控制

隔离级别: 可重复读

ID	name	age
1	张三	10

时间线	事务1	事务2
1	开启事务,分配到事务ID1	
2		开启事务,分配到事务ID2
3		修改张三的年龄为20,并且commit
4	查询张三	

结果?

张三的年龄为20

为何: 因为查询的时候才产生readview,所以事务1查询时在时间线4的时候产生的readview活跃的事务ID只有1,最小事务ID也是1,下一个分配的事务ID是3

带入到[可见性算法](# 四.可见性算法(比较的规则)) 三条全部不成立, 所以能看到

六. 案例分析2

在下面时间线之前已经存在了一条张三的记录

隔离级别: 可重复读

ID	name	age
1	张三	10

时间线	事务1	事务2
1	开启事务,分配到事务ID1	
2	查询张三的记录,当前年龄为10	
3		开启事务,分配到事务ID2
4		修改张三的年龄为20,并且commit
5	再次查询张三	

结果?

张三的年龄为10

为何: 因为 隔离级别: 可重复读 的时候, 一个事务中的多次读操作, 重复

用第一次读操作产生的readview

在时间线2的时候 产生的readview 活跃的事务ID为1和2,最小事务ID是1 下一个分配的事务ID是3

时间线5的查询的readview 还是时间线2的readview

带入到[可见性算法](# 四.可见性算法(比较的规则)) 第三条成立, 所以不看见undolog里新的那条年龄20的记录

七. 案例分析3

在下面时间线之前已经存在了一条张三的记录

隔离级别: 读已提交

ID	name	age
1	张三	10

时间线	事务1	事务2
1	开启事务,分配到事务ID1	
2	查询张三的记录,当前年龄为10	
3		开启事务,分配到事务ID2
4		修改张三的年龄为20,并且commit
5	再次查询张三	

和案例分析2唯一的区别就是隔离级别不一样

结果?

张三年龄为20

为何: 因为隔离级别: 读已提交 和 隔离级别: 可重读 的区别就在于, 隔离级别: 读已提交 每次查询都会创建1个的readview,时间线2和时间线5 产生了2个readview

八. 案例分析4

在下面时间线之前已经存在了一条张三的记录

 隔离级别: 可重复读

ID	name	age
1	张三	10

时间线	事务1	事务2
1	开启事务,分配到事务ID1	
2	查询李四(查询不到)	
3		开启事务,分配到事务ID2
4		插入李的记录,并且commit
5	查询李四	

结果?

查询不到李四

为何: 因为 隔离级别: 可重复读 的时候, 一个事务中的多次读操作, 重复用第一次读操作产生的readview

在时间线2的时候 产生的readview 活跃的事务ID为1和2,最小事务ID是1 下一个分配的事务ID是3

时间线5的查询的readview 还是时间线2的readview

带入到[可见性算法](# 四.可见性算法(比较的规则)) 第三条成立, 所以不看见undolog里新的那条李四的记录

目前看起来好像解决了幻读问题

八. 其他注意点

MVCC并没有完全解决幻读, 只是在快照读的时候解决了幻读

时间线	事务1	事务2
1	开启事务,分配到事务ID1	
2	查询李四(查询不到)	
3		开启事务,分配到事务ID2
4		插入李的记录,并且commit
5		修改李四的龄为30 或者 全表修改年龄为30
5		使用select * from user for update 语句
6	查询李四(时间线5发生的话 就能查询到数)	

注意时间线5 ,有2个时间线5,当中任何1个发生,都会使时间线6 能查询到结果,当事务中有修改操作 或者手动使用当前读之后(必须包含该记录,where条件能覆盖到新插入的数据)之后的查询会重新生成readview