事务隔离级别总结 实际情况演示 脏读(读未提交) 避免脏读(读已提交)

> 不可重复读 可重复读

幻读 参考

## Java 面试指南 / MySQL事务隔离级别详解

## MySQL事务隔离级别详解

本文由 SnailClimb 型和 guang19 型共同完成。

Quide 器数据库 ♥ MysqL = 2021年11月6日 ♦ 约 1461字

关于事务基本概览的介绍,请看这篇文章的介绍: MySQL 常见知识点&面试题总结

## SQL 标准定义了四个隔离级别:

事务隔离级别总结

• READ-UNCOMMITTED(读取未提交): 最低的隔离级别,允许读取尚未提交的数据变更,可能会导致脏读、幻读或不可重复读。

• READ-COMMITTED(读取已提交): 允许读取并发事务已经提交的数据,可以阻止脏读,但是幻读或不可重复读仍有可能发生。 • REPEATABLE-READ(可重复读): 对同一字段的多次读取结果都是一致的,除非数据是被本身事务自己所修改,可以阻止脏读和不 可重复读,但幻读仍有可能发生。

SERIALIZABLE(可串行化): 最高的隔离级别,完全服从 ACID 的隔离级别。所有的事务依次逐个执行,这样事务之间就完全不可 能产生干扰,也就是说,该级别可以防止脏读、不可重复读以及幻读。

隔离级别 脏读 不可重复读 幻读 READ-UNCOMMITTED READ-COMMITTED  $\sqrt{\phantom{a}}$ 

RI	EPEATABLE-READ	×	×	$\int$		
	SERIALIZABLE	×	×	×		
	L InnoDB 存储引擎的 MySQL 8.0 该命令改				<b>LE-READ(可重读)</b> 。我们可以通过 SELECT @@tx_isolation tion;	;命令来
1 2	MySQL> SELECT @@t	7	ion;			<u>l</u> li
3	@@tx isolation					

REPEATABLE-READ

```
从上面对 SQL 标准定义了四个隔离级别的介绍可以看出,标准的 SQL 隔离级别定义里,REPEATABLE-READ(可重复读)是不可以防止
但是!InnoDB 实现的 REPEATABLE-READ 隔离级别其实是可以解决幻读问题发生的,主要有下面两种情况:
```

当前读: 使用 Next-Key Lock 进行加锁来保证不出现幻读,Next-Key Lock 是行锁(Record Lock)和间隙锁(Gap Lock)的结 【 合,行锁只能锁住已经存在的行,为了避免插入新行,需要依赖<mark>间隙锁。</mark>

因为隔离级别越低,事务请求的锁越少,所以大部分数据库系统的隔离级别都是 READ-COMMITTED ,但是你要知道的是 InnoDB 存 储引擎默认使用 REPEATABLE-READ 并不会有任何性能损失。

实际情况演示

使用命令: START TRANSACTION 。

✓ 快照读: 由 MVCC 机制来保证不出现幻读。

InnoDB 存储引擎在分布式事务的情况下一般会用到 SERIALIZABLE 隔离级别。 《MySQL 技术内幕: InnoDB 存储引擎(第 2 版)》7.7 章这样写到:

InnoDB 存储引擎提供了对 XA 事务的支持,并通过 XA 事务来支持分布式事务的实现。分布式事务指的是允许多个独立的事务资源 (transactional resources)参与到一个全局的事务中。事务资源通常是关系型数据库系统,但也可以是其他类型的资源。全局事务

要求在其中的所有参与的事务要么都提交,要么都回滚,这对于事务原有的 ACID 要求又有了提高。另外,在使用分布式事务时,

InnoDB 存储引擎的事务隔离级别必须设置为 SERIALIZABLE。

在下面我会使用 2 个命令行 MySQL ,模拟多线程(多事务)对同一份数据的脏读问题。

我们可以通过下面的命令来设置隔离级别。 1 SET [SESSION|GLOBAL] TRANSACTION ISOLATION LEVEL [READ UNCOMMITTED|READ COMMITTED|REPEATABLE READ|SERTALIZAL

> @@TRANSACTION ISOLATION REPEATABLE-READ

> > START TRANSACTION;

.00 sec)

START TRANSACTION; OK 0 rows affected (0 00 sec) 4 第二个事务开启

mysql> use jdbcTemplate;

Database changed mysql> START TRANSACTION;

mysql> commit

mysql> s

Query OK, 0 rows affected (0.00 sec)

mysql> COMMIT; Query OK, O rows affected (0.19 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE employ SET salary=4500 WHERE id= Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0

vsql UPDATE employ SET salary 4500 WHERE id=1;

DK, 0 rows af ected (0.19 sac) 提交事务

ows matched: 1 Changed: 1 Warnings: 0

mysql> Query DK, O ro

4 第二个事务开启

5 更改数

据,但未

Q 🔡 😭 🍘

提交

MySQL 命令行的默认配置中事务都是自动提交的,即执行 SQL 语句后就会马上执行 COMMIT 操作。如果要显式地开启一个事务需要

脏读(读未提交) mysql) SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

我们再来看一下我们在下面实际操作中使用到的一些并发控制语句:

nysql> SELECT @@TRANSACTION\_ISOLATION;

• START TRANSACTION | BEGIN : 显式地开启一个事务。 • COMMIT: 提交事务, 使得对数据库做的所有修改成为永久性。

- 1 @@TRANSACTION\_ISOLATION
- 设置事务隔离级别 READ-UNCOMMITTED row in set (0.00 sec) mysql> UPDATE emplaoy SET salary=4500 WHERE id=1; ERROR 1146 (42502): Table 'idbctemplate emplaoy' mysql> UPDATE employ SET salary=4500 WHERE id=1; Query 0K, 1 row affected (0.02 sec) Rows matched: 1 Changed: 1 Warnings: 0 2 开启事务

• ROLLBACK: 回滚会结束用户的事务,并撤销正在进行的所有未提交的修改。

## nysql> SELECT salary FROM employ WHERE id=1;

START TRANSACTION;



▶ 3 确认数据是5000

6 因为事务隔离级

不会发生脏读

别为 读已提交,所以

8 读取到的数据是

还是刚才上面的读已提交的图,虽然避免了读未提交,但是却出现了,一个事务还没有结束,就发生了 不可重复读问题。

已提交的数据

### nysql> SELECT salary FROM employ WHERE id = 1 salary

row in set (0.00 sec)

row in set (0.03 sec)

salary 5000

5000

salary

不可重复读

READ-COMMITTED

salary 5000

salary

l row in set (0.00 sec)

row in set (0 00 sec)

nysql> START TRANSACTION;

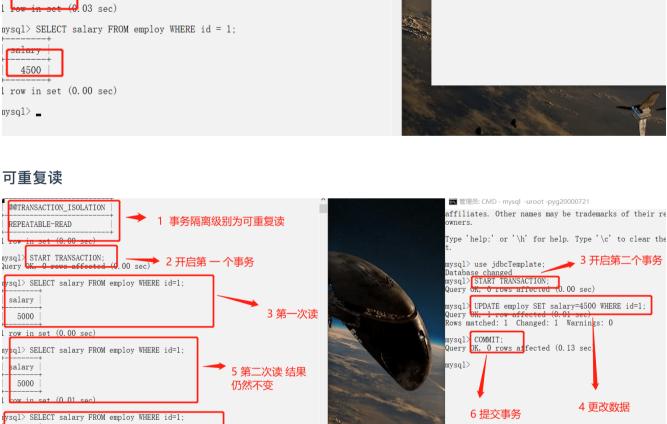
nysql> SELECT salary FROM employ WHERE id = 1;

nysql> SELECT salary FROM employ WHERE id = 1;

ysql> SELECT salary FROM employ WHERE id = 1

nysql> SELECT salary FROM employ WHERE id = 1

5000 . 03 sec)



7 第三次读

结果仍然不变

# 幻读

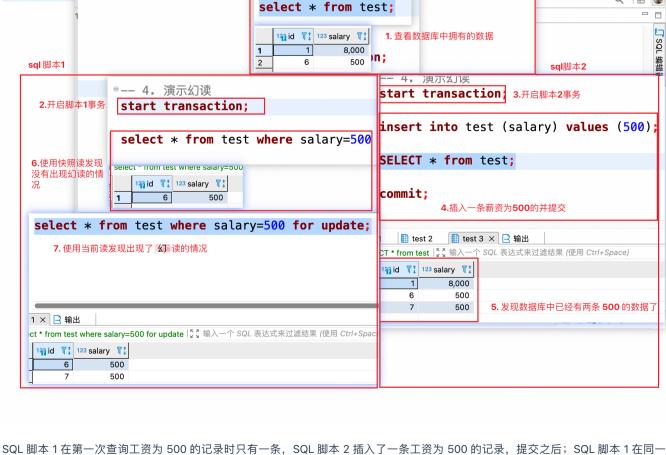
salary

5000

nysql> ^Zs

row in set (0.00 sec)

演示幻读出现的情况



个事务中再次使用当前读查询发现出现了两条工资为 500 的记录这种就是幻读。

←1. 将事务隔离级别调整为 SERIALIZABLE 。 2. 在可重复读的事务级别下,给事务操作的这张表添加表锁。 3. 在可重复读的事务级别下,给事务操作的这张表添加 Next-key Lock (Record Lock+Gap Lock) 。

- 参考
- Mysql 锁: 灵魂七拷问♂ ● Innodb 中的事务隔离级别和锁的关系 ゼ

🗹 编辑此页 🖒 上次编辑于: 2022/7/10 20:11:46 贡献者: guide,wangtong,Guide,sam,yueyang

〈上一页

InnoDB存储引擎对MVCC的实现

解决幻读的方法 解决幻读的方式有很多,但是它们的核心思想就是一个事<u>务在操作某张表数据的时候,</u>另外一个事务不允许新增或者删除这张表中的 数据了。解决幻读的方式主要有以下几种:

• 《MySQL 技术内幕: InnoDB 存储引擎》

https://dev.MySQL.com/doc/refman/5.7/en/☑

MySQL三大日志(binlog、redo log和undo log)详解 鄂ICP备2020015769号-1 Copyright © 2022 Guide



 $\perp$ 

## 快速理解脏读、不可重复读、幻读

发布于2019-06-25 17:49:12 阅读 36.3K

脏漠、不可单复海、幻姿指的一个转在批介的 遇到的数据问题

项目技术点

理解这三种由于并发访问导致的数据读取问题,再理解事务隔离级别就简单多了 脏读 (读取未提交数据):一个事务在执行与这些广泛取了共石事务术提及公数据

A事务读取B事务尚未提交的数据,此时如果B事务发生错误并执行回滚操作,那么A事务读取到的数据就是脏数据。就 好像原本的数据比较干净、纯粹,此时由于B事务更改了它,这个数据变得不再纯粹。这个时候A事务立即读取了这个脏 数据,但事务B良心发现,又用回滚把数据恢复成原来干净、纯粹的样子,而事务A却什么都不知道,最终结果就是事务 A读取了此次的脏数据,称为脏读。

这种情况常发生于转账与取款操作中

时间顺序	转账事务	取款事务
1		开始事务
2	开始事务	
3		查询账户余额为2000元
4		取款1000元,余额被更改为1000 元
5	查询账户余额为1000元 (产生脏读)	
6		取款操作发生未知错误,事务回滚,余额变更为2000元
7	转入2000元,余额被更改为3000 元 (脏读的1000+2000)	
8	提交事务	
备注	按照正确逻辑, 此时账户余额应该	为4000元

# 不可重复读(前后多次读取,数据内容不一致):一个一种在内外的过程中分次使取同一数据的内容不一致

事务A在执行读取操作,由整个事务A比较大,前后读取同一条数据需要经历很长的时间 。而在事务A第一次读取数据, 比如此时读取了小明的年龄为20岁,事务B执行更改操作,将小明的年龄更改为30岁,此时事务A第二次读取到小明的 年龄时,发现其年龄是30岁,和之前的数据不一样了,也就是数据不重复了,系统不可以读取到重复的数据,成为不可 重复读。

时间顺序	事务A	事务B	
1	开始事务		
2	第一次查询,小明的年龄为20岁		
3		开始事务	
4	其他操作		
5		更改小明的年龄为30岁	
6		提交事务	
7	第二次查询,小明的年龄为30岁		
备注	按照正确逻辑,事务A前后两次读取到的数据应该一致		

# 幻读 (前后多次读取,数据总量不一致):一个整在执行的比较少的次所读取的总量不一致

事务A在执行读取操作,需要两次统计数据的总量,前一次查询数据总量后,此时事务B执行了新增数据的操作并提交 后,这个时候事务A读取的数据总量和之前统计的不一样,就像产生了幻觉一样,平白无故的多了几条数据,成为幻 读。

时间顺序	事务A	事务B	
1	开始事务		
2	第一次查询,数据总量为100条		
3		开始事务	
4	其他操作		
5		新增100条数据	
6		提交事务	
7	第二次查询,数据总量为200条		
备注	按照正确逻辑,事务A前后两次读取到的数据总量应该一致		

不可重复读和幻读到底有什么区别呢?

解决:使用行级锁,锁定该行,事务A多次读取操作完成后才释放该锁,这个时候才允许其他事务更改刚才的数据。 (2) 幻读是读取了其他事务新增的数据,针对insert和delete操作

(1) 不可重复读是读取了其他事务更改的数据,针对update操作

解决:使用表级锁,锁定整张表,事务A多次读取数据总量之后才释放该锁,这个时候才允许其他事务新增数据。

这时候再理解事务隔离级别就简单多了呢。