21 不同事务隔离级别有哪些区别?

更新时间：2019-09-30 11:10:28

> 天才就是长期劳动的结果。
>
> ——牛顿

在之前的几节中，我们都提到了事务隔离级别，但是都只是大致地过了一遍，本节就来深入探讨一下。

1 通过基本定义认识事务隔离级别

MySQL 有四种隔离级别，我们来看一下这四种隔离级别的基本定义：

- Read uncommitted（读未提交，简称：RU）：在该隔离级别，所有事务都可以看到其它未提交的事务的执行结果。可能会出现脏读。
- Read Committed（读已提交，简称： RC）：一个事务只能看见已经提交事务所做的改变。因为同一事务的其它实例在该实例处理期间可能会有新的 commit，所以可能出现幻读。
- Repeatable Read（可重复读，简称：RR）：这是 MySQL 的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行。消除了脏读、不可重复读，默认也不会出现幻读。
- Serializable（串行）：这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。

2 通过实验认识事务隔离级别

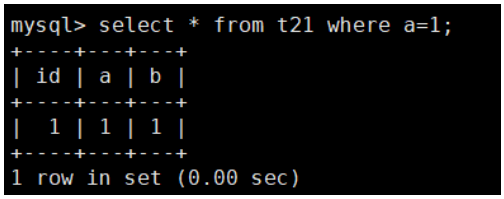
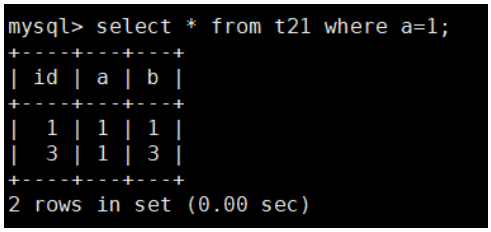
为了便于理解事务隔离级别，这里通过几个实验理解一下各个隔离级别的特性。

首先创建测试表并写入测试数据，语句如下：

```
use muke;

drop procedure if exists insert_t21; /* 如果存在存储过程insert_t21，则删除 */
delimiter ;;
create procedure insert_t21() /* 创建存储过程insert_t21 */
begin

drop table if exists t21;

CREATE TABLE `t21` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`a` int(11) NOT NULL,
`b` int(11) NOT NULL,
PRIMARY KEY (`id`),
KEY `idx_c` (`a`)
) ENGINE=InnoDB CHARSET=utf8mb4;
insert into t21(a,b) values (1,1),(2,2);

end;;
delimiter ; /* 存储过程insert_t21主要功能创建测试表 t21，并写入数据 */
```

下面我们开始进行事务隔离级别实验：

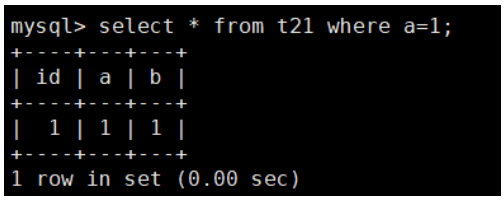
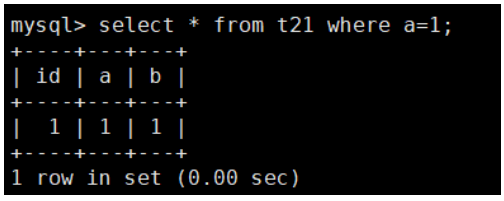
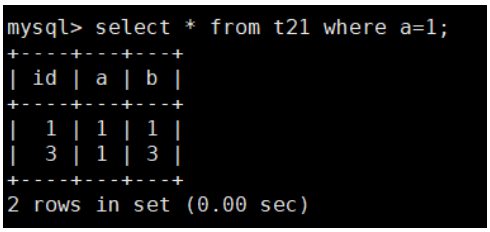
2.1 Read uncommitted 实验

ID	session1	session2
1	call insert_t21(); /* 运行存储过程 insert_t21 */	
2	set session transaction_isolation='READ-UNCOMMITTED';	set session transaction_isolation='READ-UNCOMMITTED';
3	begin;	begin;
4	select * from t21 where a=1; 	
5		insert into t21(a,b) values (1,3);
6	select * from t21 where a=1; 	
7	commit;	commit;

上面的实验中，第 5 步中 session2 写入了一条 a、b 值分别为 1、3 的记录，在第 6 步中，session2 中的事务还没提交，但是 session1 就能看到 session2 写入的数据，出现脏读现象。

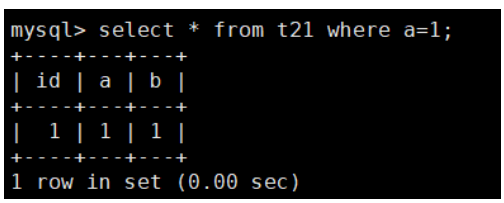
2.2 Read Committed 实验

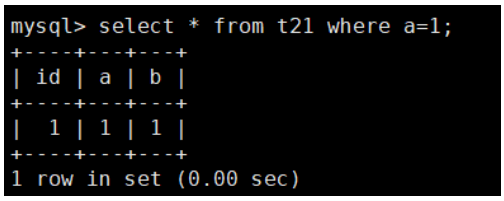
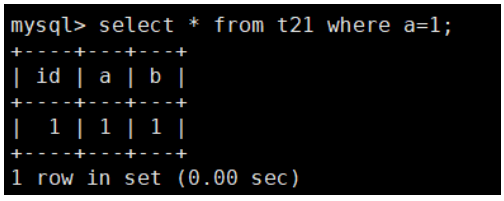
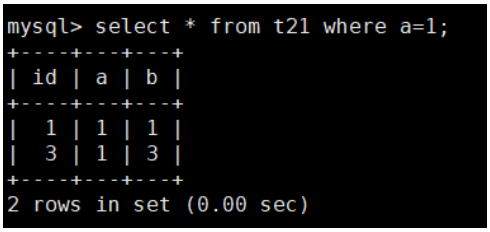
ID	session1	session2
1	call insert_t21 (); /* 运行存储过程 insert_t21 */	
2	set session transaction_isolation='READ-COMMITTED';	set session transaction_isolation='READ-COMMITTED';
3	begin;	begin;

ID	session1	session2
4	select * from t21 where a=1; 	
5		insert into t21(a,b) values (1,3);
6	select * from t21 where a=1; 	
7		commit;
8	select * from t21 where a=1; 	
9	commit;	

session2 写入了新数据未提交的情况下，session1 无法查看到新记录，等到 session2 提交之后，session1 才能看到第 5 步 session2 写入的数据。

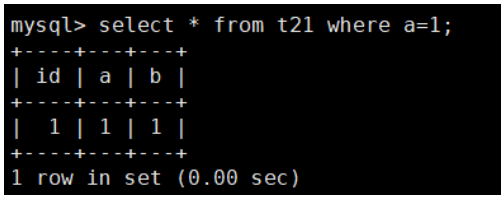
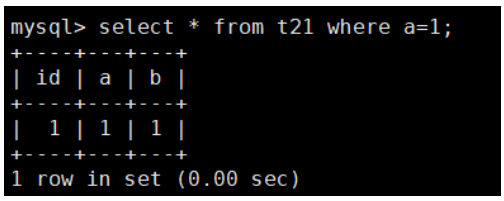
2.3 Repeatable Read 实验

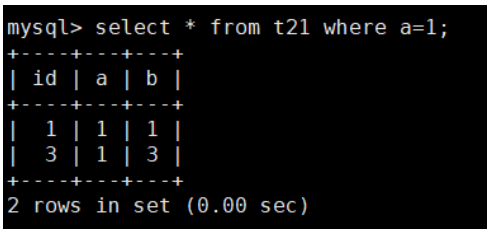
ID	session1	session2
1	call insert_t21 (); /* 运行存储过程 insert_t21 */	
2	set session transaction_isolation='REPEATABLE-READ';	set session transaction_isolation='REPEATABLE-READ';
3	begin;	begin;
4	select * from t21 where a=1; 	
5		insert into t21(a,b) values (1,3);

ID	session1	session2
6	select * from t21 where a=1; 	
7		commit;
8	select * from t21 where a=1; 	
9	commit;	
10	select * from t21 where a=1; 	

session2 写入了新数据未提交的情况下，session1 无法查看到新记录，等到 session2 提交但是 session1 还未提交时，session1 还是不能看到新记录，需要等 session1 事务提交之后，才能查看到第 5 步 session2 写入的新数据。

2.4 Serializable 实验

ID	session1	session2
1	call insert_t21 (); /* 运行存储过程 insert_t21 */	
2	set session transaction_isolation='SERIALIZABLE';	set session transaction_isolation='SERIALIZABLE';
3	begin;	begin;
4	select * from t21 where a=1; 	
5		insert into t21(a,b) values (1,3); （等待）
6	select * from t21 where a=1; 	
7	commit;	session1 提交后，第 5 步中的写入操作执行成功
8		commit;

ID	session1	session2
9	select * from t21 where a=1;  	

当 session1 中有事务查询 a=1 这行记录时，在 session2 就不能插入 a=1 的记录，进入等待。必须等 session1 提交后，session2 才能执行成功。也就是让事务串行进行。

3 通过生活中的例子认识事务隔离级别

3.1 Read uncommitted 的例子

拿零售业务场景来讲，在事务隔离级别 RU 下：比如顾客 A 在超市买单时，当收银员扫完顾客 A 的支付码后，因为网络原因，一直等待着（也就是整个支付过程的事务还没结束）；这时收银员去后台数据查询，看到 A 的钱已经进入超市账户了，然后让顾客 A 离开。过了一会，整个支付过程回滚了，才发现 A 实际是支付失败。这样超市岂不是很亏。这就是 RU 隔离级别可能导致脏读的情况。

3.2 Read Committed 的例子

在 RC 隔离级别下：比如顾客 A 在超市购买了 90 元的东西，当收银系统查询到顾客 A 还剩 100 元，足够扣款，此时 A 的老婆在家网购，花掉了 A 账户里的这 100 块，这时收银系统在扣除 A 账户 90 元这一步操作时，就会出现报错的情况。这时顾客 A 肯定郁闷，不是明明钱够么？这就是 RC 隔离级别下的幻读现象。

3.3 Repeatable Read 的例子

还是拿上面的例子，顾客 A 在超市购买了 90 元的东西，当收银系统查询到顾客 A 还剩 100 元，足够扣款，此时 A 的老婆在家网购，能查询到 A 的账户里还有 100 元，但是想要用 A 账户里的 100 块，却发现并不能使用这 100 元。这样，A 最后的扣款步骤也能正常完成，最终顺利完成了整个付款过程。这就是可重复读的现象。

3.4 Serializable 的例子

顾客 A 在超市购买了 90 元的东西，当收银系统查询到顾客 A 还剩 100 元，足够扣款，此时 A 的老婆在家网购，想查询 A 账户里还有多少钱，却发现无法查看到，必须要等到 A 整个付款完成，其老婆才能去查询余额。这就是串行导致的。

4 如何选择合适的事务隔离级别

在上面的内容中，我们认识了事务隔离级别，那么应该怎样选择合适的事务隔离级别呢？

对于 RU 隔离级别，会导致脏读，从性能上看，也不会比其它隔离级别好太多，因此生产环境不建议使用。

对于 RC 隔离级别，相比 RU 隔离级别，不会出现脏读；但是会出现幻读，一个事务中的两次执行同样的查询，可能得到不一样的结果。

对于 RR 隔离级别，相比 RC 隔离级别，不会出现幻读（这个在第 17 节详细讲了，RR 隔离级别通过间隙锁解决了幻读），但是相对于 RC，锁的范围可能更大了。

对于 Serializable 隔离级别，因为它强制事务串行执行，会在读取的每一行数据上都加锁，因此可能会导致大量的超时和锁争用的问题。生成环境很少使用。

因此总的来说，建议在 RC 和 RR 两个隔离级别中选一种，如果能接受幻读，需要并发高点，就可以配置成 RC，如果不能接受幻读的情况，就设置成 RR 隔离级别。

5 总结

本节讲解了事务隔离级别，MySQL 的事务隔离级别分为 4 种：

- Read uncommitted（读未提交，简称：RU）
- Read Committed（读已提交，简称： RC）
- Repeatable Read（可重复读，简称：RR）
- Serializable（串行）

并通过实验验证了几种隔离级别的特点。

对于选择隔离级别的建议如下：

建议在 RC 和 RR 两个隔离级别中选一种，如果能接受幻读，需要并发高点，就可以配置成 RC，如果不能接受幻读的情况，就设置成 RR 隔离级别。

6 问题

ID	session1	session2
1	call insert_t21 (); /* 运行存储过程 insert_t21 */	
2	set session transaction_isolation='READ-COMMITTED'; or set session transaction_isolation='REPEATABLE-READ';	set session transaction_isolation='READ-COMMITTED'; or set session transaction_isolation='REPEATABLE-READ';
3	begin;	begin;
4	select * from t21 where a=1; R1	
5		update t21 set b=3 where a=1;
6	select * from t21 where a=1; R2	
7		commit
8	select * from t21 where a=1; R3	
9	commit;	
10	select * from t21 where a=1; R4	

RC 隔离级别和 RR 隔离级别下 R1-R4 的结果分别是？可以先观察写下自己的答案，然后通过实验验证自己的答案是否正确，也欢迎把你的结果分享在留言区。

7 参考资料

《高性能 MySQL》第 3 版：1.3.1 隔离级别

}