事务隔离级别与IO的关系

课程内容

1. 事务类型以及序列化介绍
2. 事务隔离级别之RU级别-事务隔离级别设置
3. RU级别下存在的问题
4. 事务RC隔离级别
5. mysql事务RR级别-幻读问题
6. mysql事务解决幻读问题
7. 事务隔离级别实现
8. 事务隔离级别与IO的关系

1. 事务隔离级别类型以及序列化介绍

事务的隔离性是多个用户并发访问数据库时，数据库为每一个用户开启的事务，不能被其他事务的操作数据所干扰，多个并发事务之间要相互隔离。 在并发下事务会容易出现一些问题：（先不急着演示这些情况）

- 脏读 ：一个事务开始读取了某行数据，另外一个事务已经更新了此数据但没有能够及时提交。这是相当危险的，因为很可能所有的操作都被回滚。
- 不可重复读：一个事务对同一行数据重复读取两次，但是却得到了不同的结果。例如，在两次读取的中途，有另外一个事务对该型数据进行了修改，并提交。
- 幻读：事务在操作过程中进行两次查询，第二次查询的结果包含了第一次查询中未出现的数据（SQL不一定一样）。

这是因为在两次查询过程中有另外一个 事务插入数据 在MySQL中存在（InnoDB）事务存在着4中隔离级别，不同的隔离级别对事务的处理不同。

- 未授权读取（未提交读 Read Uncommitted）：READ-UNCOMMITTED | 0：存在脏读，不可重复读，幻读的问题。如果一个事务已经开始写数据，则另外一个数据则不会允许同时进行写操作，但允许其他事务读此行数据。隔离级别可以通过“排他写锁”实现。
- 授权读取（已读提交 Read committed）：READ-COMMITTED | 1：解决脏读的问题，存在不可重复读，幻读的问题。这个可以通过“排他写锁”实现。读取数据的事务允许其他事务继续访问该行数据，但是未提交的写事务将会禁止其他事务访问该行。
- 可重复读取（Repeatable Read）：REPEATABLE-READ | 2：解决脏读，不可重复读的问题，存在幻读的问题，默认隔离级别。可通过“共享锁”，“排他锁”实现。读取数据的事务将会禁止写事务（但允许读事务），写事务则禁止任何其他事务。
- 序列化（Serializable）：SERIALIZABLE | 3：解决脏读，不可重复读，幻读，可保证事务安全，但完全串行执行，性能最低。提供严格的事务隔离。它要求事务序列化执行，事务只能一个接着一个地执行，不能并发执行。如果仅仅通过“行级锁”是无法实现事务序列化的，必须要通过其他机制保证新插入的数据不会被刚执行查询操作的事务访问到。

事务隔离级别

隔离级别	读数据一致性	脏读	不可重复读	幻读
未提交读 Read Uncommitted	最低级别，只能保证不读取物理上损坏的数据	是	是	是
已读提交 Read committed	语句级	否	是	是
可重复读取 Repeatable Read	事务隔离级别	否	否	是
序列化（Serializable）	最高级别，事务级	否	否	否

2. 事务隔离级别之RU级别-事务隔离级别设置

2.1 事务隔离级别设置

可以通过 show variables like '%iso%'; 查看当前的隔离级别

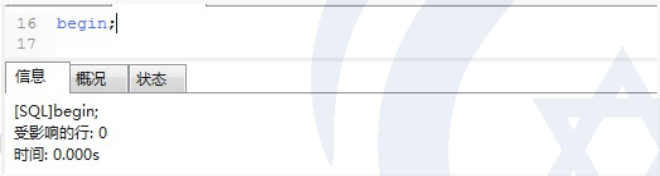
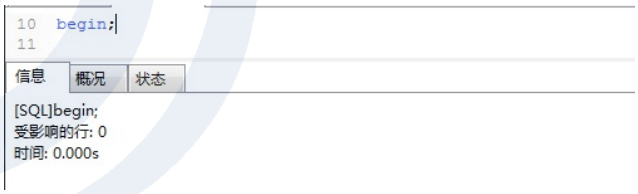
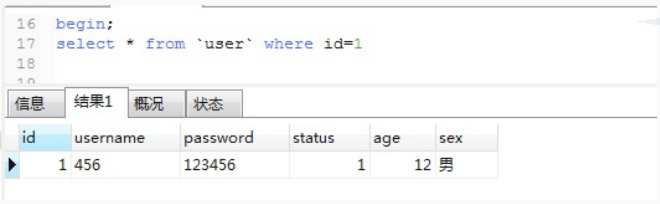
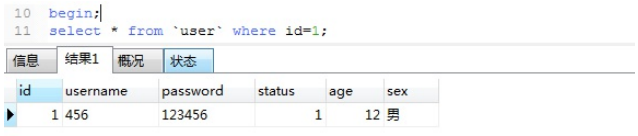
```
SHOW VARIABLES LIKE '%transaction_isolation%';--查看
+-----------------------+------------------+
| Variable_name         |     Value        |
+-----------------------+------------------+
| transaction_isolation | REPEATABLE-READ |
+-----------------------+------------------+
```

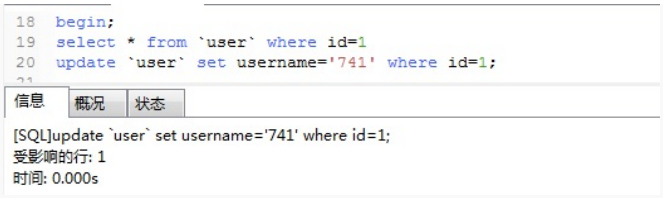
```
-- 设定全局的隔离级别 设定会话 global 替换为 session 即可 把set语法温习一下
-- SET [GLOABL] config_name = 'foobar';
-- SET @@[session|global].config_name = 'foobar';
-- SELECT @@[global.]config_name;

--全局修改
SET @@gloabl.transaction_isolation = 0;
SET @@gloabl.transaction_isolation = 'READ-UNCOMMITTED';
SET @@gloabl.transaction_isolation = 1;
SET @@gloabl.transaction_isolation = 'READ-COMMITTED';
SET @@gloabl.transaction_isolation = 2;
SET @@gloabl.transaction_isolation = 'REPEATABLE-READ';
SET @@gloabl.transaction_isolation = 3;
SET @@gloabl.transaction_isolation = 'SERIALIZABLE';

--局部修改
SET @@session.transaction_isolation = 0;
SET @@session.transaction_isolation = 'READ-UNCOMMITTED';
SET @@session.transaction_isolation = 1;
SET @@session.transaction_isolation = 'READ-COMMITTED';
SET @@session.transaction_isolation = 2;
SET @@session.transaction_isolation = 'REPEATABLE-READ';
SET @@session.transaction_isolation = 3;
SET @@session.transaction_isolation = 'SERIALIZABLE';
```

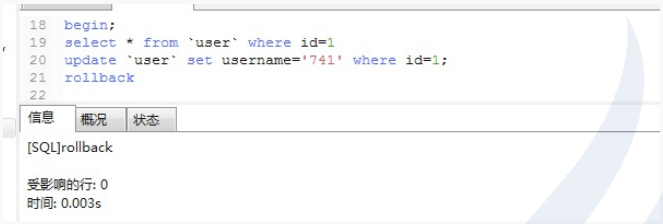
2.2 事务RU级别产生的问题

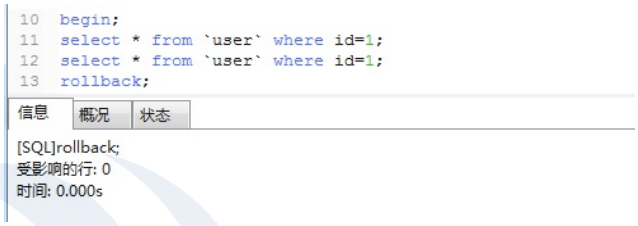
解决问题：没有使用事务时数据不一致问题，做到所有的sql一起执行，一起失败

脏读 ：一个事务开始读取了某行数据，另外一个事务已经更新了此数据但没有能够及时提交。这是相当危险的，因为很可能所有的操作都被回滚。（也会存在脏读，幻读问题）

session1	session2
	
	
	
	







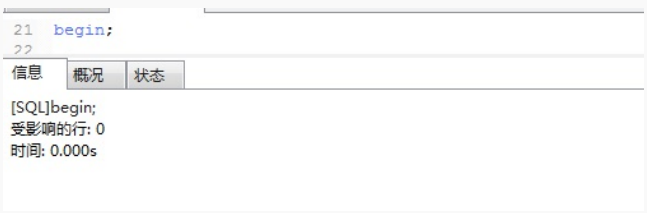
3. 事务RC隔离级别

解决问题：解决脏读的问题

不可重复读：一个事务对同一行数据重复读取两次，但是却得到了不同的结果。例如，在两次读取的中途，有另外一个事务对该型数据进行了修改并提交。

问题演示：

```
--修改事务隔离级别为RC级别
set @@global.transaction_isolation=1;
show global variables like '%transaction_isolation%';
```

session1	session2
	

```
16  begin
```

信息 | 概况 | 状态

[SQL]begin

受影响的行: 0
时间: 0.000s

```
21  begin;
22  select * from `user` where id=1;
```

信息 | 结果1 | 概况 | 状态

id	username	password	status	age	sex
1	85211	123456	1	12	男

```
16  begin
17  select * from `user` where id=1;
```

信息 | 结果1 | 概况 | 状态

id	username	password	status	age	sex
1	85211	123456	1	12	男

```
21  begin;
22  select * from `user` where id=1;
23  update `user` set username='741' where id=1;
```

信息 | 概况 | 状态

[SQL]update `user` set username='741' where id=1;
受影响的行: 1
时间: 0.001s

```
16  begin
17  select * from `user` where id=1;
18  select * from `user` where id=1;
```

信息 | 结果1 | 概况 | 状态

id	username	password	status	age	sex
1	85211	123456	1	12	男

```
21  begin;
22  select * from `user` where id=1;
23  update `user` set username='741' where id=1;
24  commit;
25
```

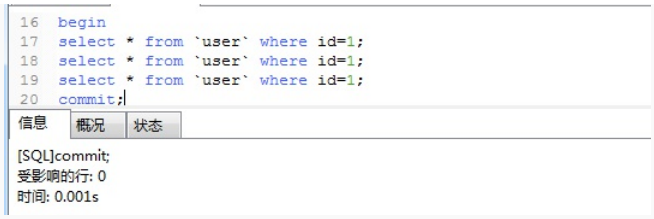
信息 | 概况 | 状态

[SQL]commit;
受影响的行: 0
时间: 0.201s

```
16  begin
17  select * from `user` where id=1;
18  select * from `user` where id=1;
19  select * from `user` where id=1;
20
```

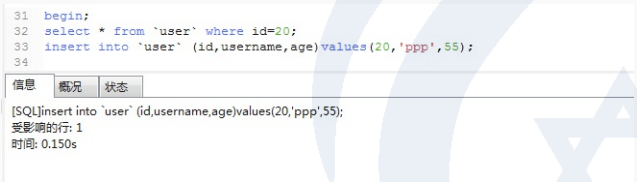
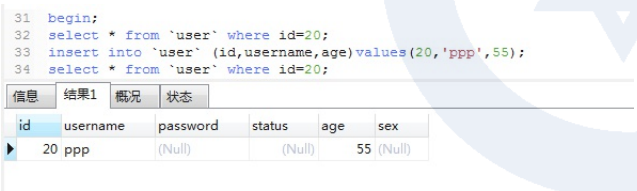
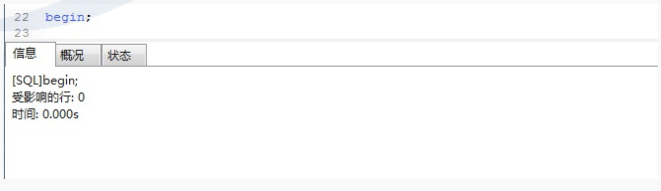
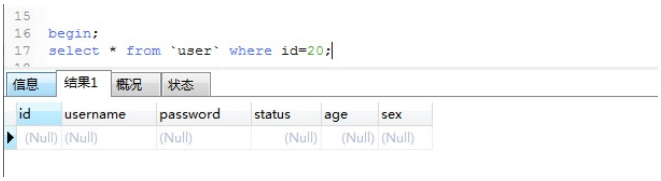
信息 | 结果1 | 概况 | 状态

id	username	password	status	age	sex
1	741	123456	1	12	男



5. mysql事务RR级别-幻读问题（mysql8已解决该问题）

幻读：事务在操作过程中进行两次查询，第二次查询的结果包含了第一次查询中未出现的数据（SQL不一定一样）。

session1	session2
	
	
	
	
	
	

```
16  begin;
17  select * from `user` where id=20;
18  insert into `user` (id,username,age)values(20,'ppp',55);
```

信息

[SQL]insert into `user` (id,username,age)values(20,'ppp',55);

6. 事务隔离级别实现

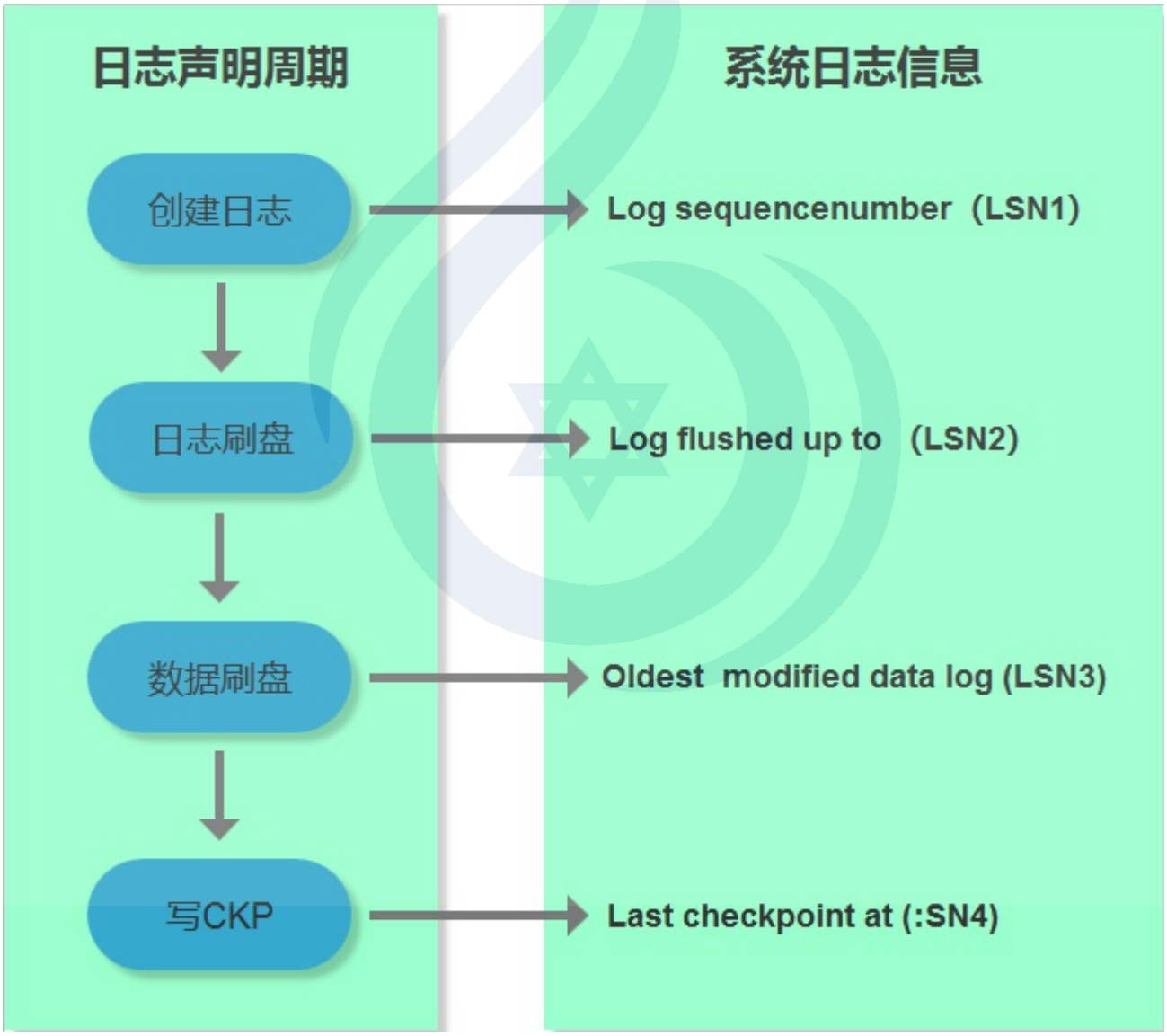
未授权提交（未提交读 Read UNCommitted）：主要在于重做和回滚-共享锁 在事务进行中，允许其他事务可以读取修改的数据。

授权提交（）：使用了排它锁，不允许其他事务读取到数据，同时也不能修改

可重复读取：排它锁，会把可读取的数据放在缓存区中，而其他事务修改的数据会加一把锁

序列化：串行化，实现不仅仅只是靠行锁，还会加上其他机制实现

7. 事务隔离级别与IO的关系



Innodb的一条事务日志共经历4个阶段：

```
1）创建阶段：事务创建一条日志；
2）日志刷盘：日志写入到磁盘上的日志文件；  （ib_logfile里面）
```

```
3）数据刷盘：日志对应的脏页数据写入到磁盘上的数据文件；
4）写CKP：日志被当作Checkpoint写入日志文件；（ib_data里面）
```

7.1 概述

```
-- 查看日志文件设置状态
show variables like 'innodb_%';
-- 更改
set @@global.innodb_flush_log_at_trx_commit = 0; -- 0, 1, 2
show variables like 'innodb_flush_log_at_trx_commit';
```