Database Concepts lab 7-8

Lining

2020/10/31

1. Exp 7 - Transaction commands

Functions	Examples
view current active transaction	select * from information_schema.innodb_trx;

set/view auto commit	set @@autocommit=0; set global @@autocommit=0;
	set autocommit = 0; set global autocommit = 0;
	show global/session variables like 'autocommit';
	(when omit global/session, the range of session is default value)

set/view global isolation level	set global transaction isolation level repeatable read; show global variables like 'transaction_isolation';
set/view current session isolation level	set transaction isolation level repeatable read; set session transaction isolation level repeatable read;

	show variables like 'transaction_isolation';
	show session variables like 'transaction_isolation';
	select @@transaction_isolation;

1. Exp 7 - View lock

- Show engine innodb status
- Three tables: innodb_locks, innodb_trx, innodb_lock_waits.

```
select * from information_schema.innodb_trx
select * from performance_schema.data_locks
select * from sys.innodb_lock_waits;
```

Table innodb_locks

Column Name	DISCRIPTION
LOCK_ID	A unique lock ID number,
LOCK_TRX_ID	The ID of the transaction holding the lock.
LOCK_MODE	How the lock is requested. Permitted lock mode descriptors are S, X, IS, IX, GAP, AUTO_INC, and UNKNOWN.
LOCK_TYPE	The type of lock. RECORDfor a row-level lock,TABLEfor a table-level lock.
LOCK_TABLE	The name of the table that has been locked or contains locked records.
LOCK_INDEX	The name of the index, ifLOCK_TYPEisRECORD; otherwiseNULL.
LOCK_SPACE	The tablespace ID of the locked record, ifLOCK_TYPEisRECORD; otherwiseNULL.
LOCK_SPACE	The page number of the locked record, ifLOCK_TYPEisRECORD; otherwiseNULL.
LOCK_SPACE	The heap number of the locked record within the page, ifLOCK_TYPEisRECORD; otherwiseNULL.
LOCK_SPACE	The data associated with the lock. A value is shown if the LOCK_TYPE is RECORD, otherwise the value is NULL

1. Exp 7 - View locks

■How to view locks

Table innodb_trx (information of transaction, no tables, trx_id available)

Column Name	DISCRIPTION
TRX_ID	A unique transaction ID number, internal to InnoDB. These IDs are not created for transactions that are read only and nonlocking.
TRX_WEIGHT	The weight of a transaction, reflecting (but not necessarily the exact count of) the number of rows altered and the number of rows locked by the transaction.
TRX_STATE	The transaction execution state. Permitted values areRUNNING,LOCK WAIT,ROLLING BACK, andCOMMITTING.
TRX_STARTED	The transaction start time.
TRX_REQUESTED_LOCK_ID	The ID of the lock the transaction is currently waiting for, if TRX_STATE is LOCK WAIT; otherwise NULL.
TRX_WAIT_STARTED	The time when the transaction started waiting on the lock, ifTRX_STATEisLOCK WAIT; otherwiseNULL.
TRX_MYSQL_THREAD_ID	The MySQL thread ID.
TRX_QUERY	The SQL statement that is being executed by the transaction.

Table innodb_lock_waits

Column Name	DISCRIPTION
REQUESTING_TRX_ID	The ID of the requesting (blocked) transaction.
REQUESTED_LOCK_ID	The ID of the lock for which a transaction is waiting.
BLOCKING_TRX_ID	The ID of the blocking transaction.
BLOCKING_LOCK_ID	The ID of a lock held by a transaction blocking another transaction from proceeding.

1. Exp 7 - View locks

■How to view locks(2)

Table innodb_lock_waits

Column Name	DISCRIPTION
REQUESTING_TRX_ID	The ID of the requesting (blocked) transaction.
REQUESTED_LOCK_ID	The ID of the lock for which a transaction is waiting.
BLOCKING_TRX_ID	The ID of the blocking transaction.
BLOCKING_LOCK_ID	The ID of a lock held by a transaction blocking another transaction from proceeding.

1. Exp 7 - Detail information

西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

1. Details for table innodb_locks

   https://dev.mysql.com/doc/refman/5.7/en/information-schema-innodb-locks-table.html

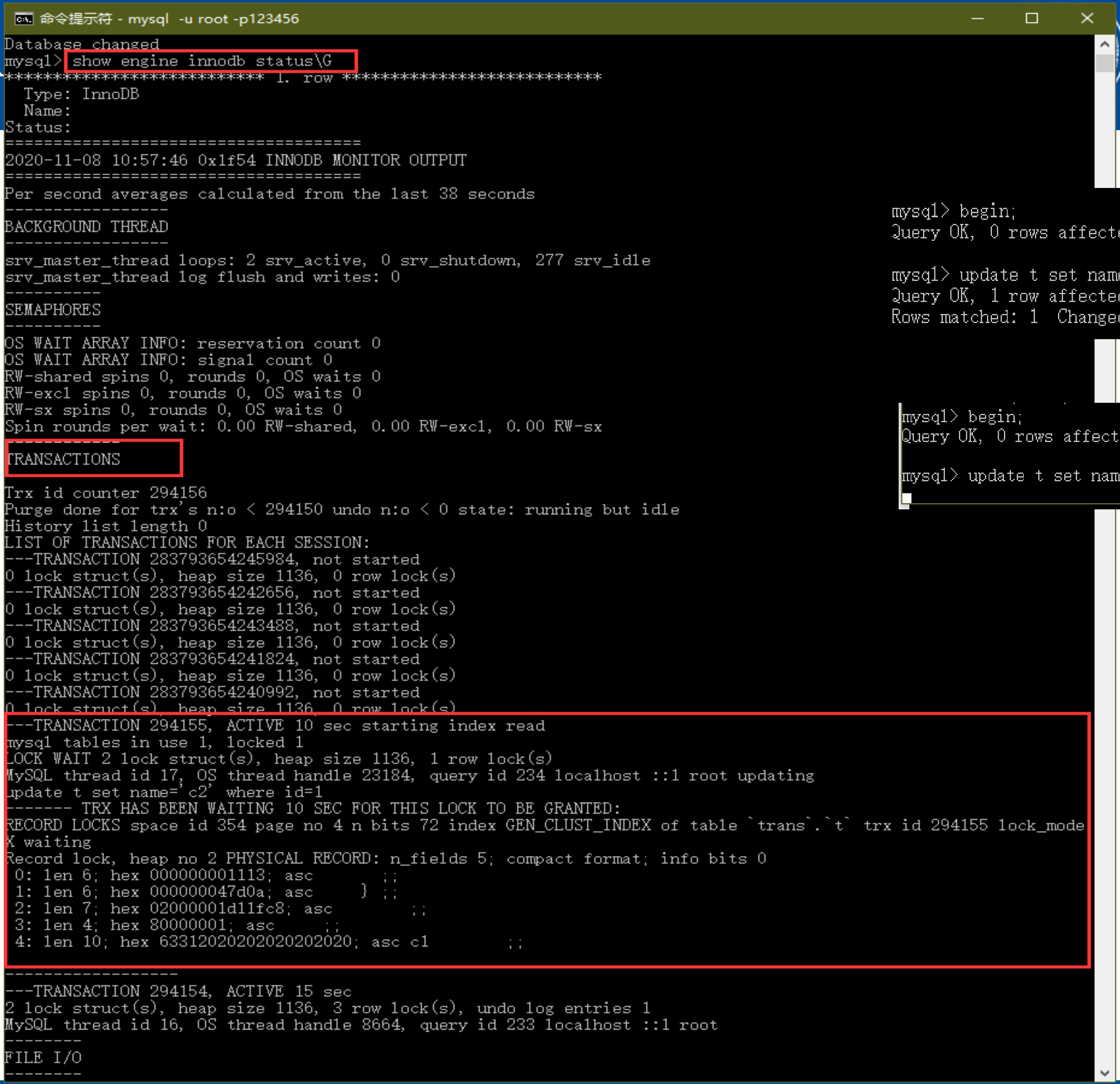
2. Details for table innodb_trx

   https://dev.mysql.com/doc/refman/5.7/en/information-schema-innodb-trx-table.html

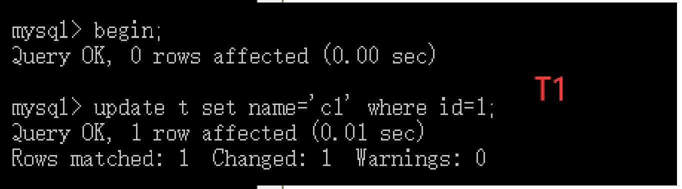
3. Details for table innodb_trx

   https://dev.mysql.com/doc/refman/5.7/en/information-schema-innodb-lock-waits-table.html

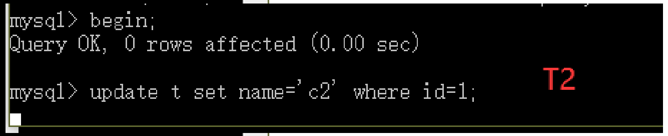
1



```
命令提示符 - mysql  -u root -p123456
Database changed
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2020-11-08 10:57:46 0x1f54 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 38 seconds
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 2 srv_active, 0 srv_shutdown, 277 srv_idle
srv_master_thread log flush and writes: 0
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 0
OS WAIT ARRAY INFO: signal count 0
RW-shared spins 0, rounds 0, OS waits 0
RW-excl spins 0, rounds 0, OS waits 0
RW-sx spins 0, rounds 0, OS waits 0
Spin rounds per wait: 0.00 RW-shared, 0.00 RW-excl, 0.00 RW-sx
------------
TRANSACTIONS
------------
Trx id counter 294156
Purge done for trx's n:o < 294150 undo n:o < 0 state: running but idle
History list length 0
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 283793654245984, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 283793654242656, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 283793654243488, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 283793654241824, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 283793654240992, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 294155, ACTIVE 10 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 17, OS thread handle 23184, query id 234 localhost ::1 root updating
update t set name='c2' where id=1
------- TRX HAS BEEN WAITING 10 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 354 page no 4 n bits 72 index GEN_CLUST_INDEX of table `trans`.`t` trx id 294155 lock_mode
X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
 0: len 6; hex 000000001113; asc       ;;
 1: len 6; hex 000000047d0a; asc     } ;;
 2: len 7; hex 02000001d11fc8; asc        ;;
 3: len 4; hex 80000001; asc     ;;
 4: len 10; hex 63312020202020202020; asc c1        ;;

------------------
---TRANSACTION 294154, ACTIVE 15 sec
2 lock struct(s), heap size 1136, 3 row lock(s), undo log entries 1
MySQL thread id 16, OS thread handle 8664, query id 233 localhost ::1 root
--------
FILE I/O
--------
```

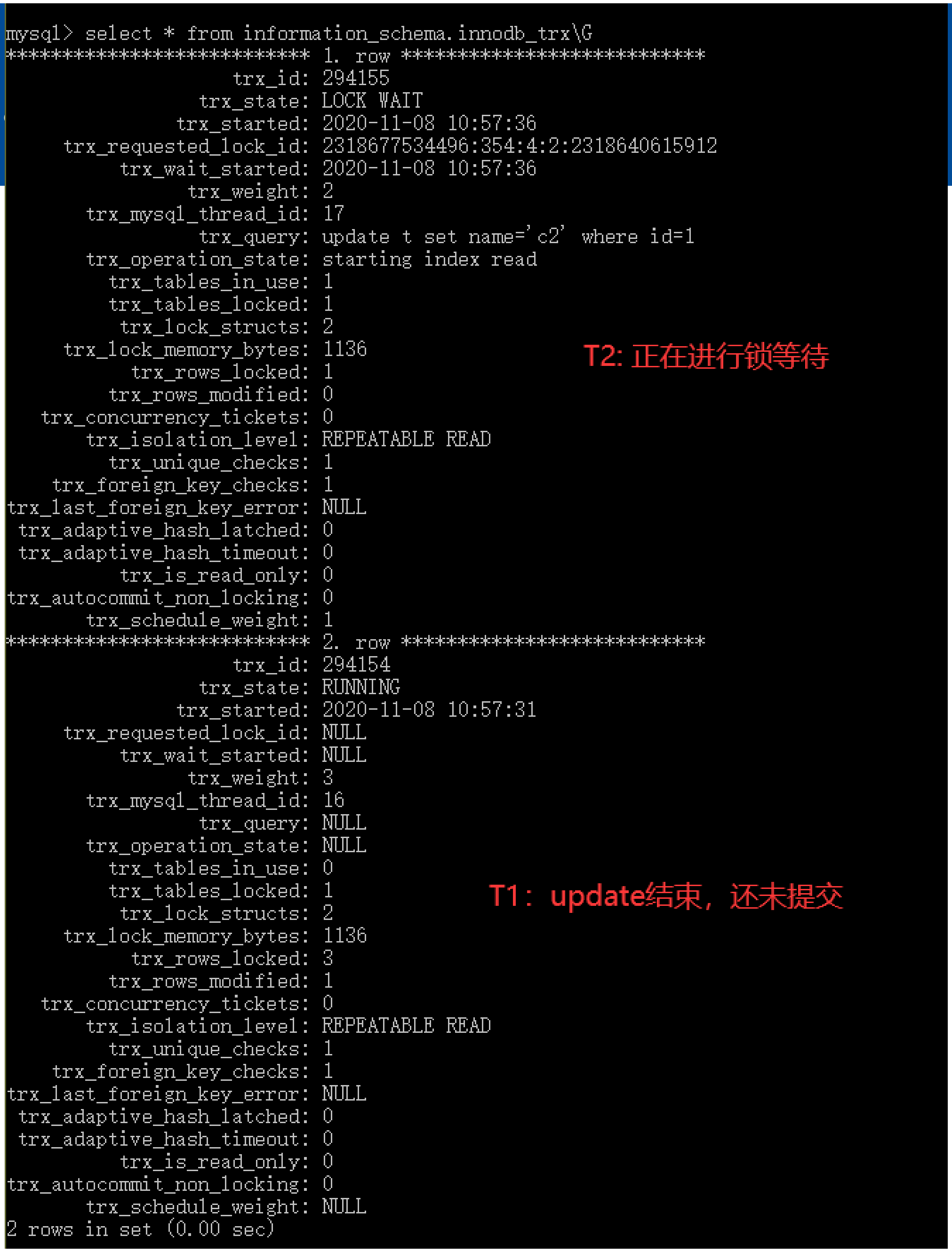


```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> update t set name='c1' where id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> update t set name='c2' where id=1;
```

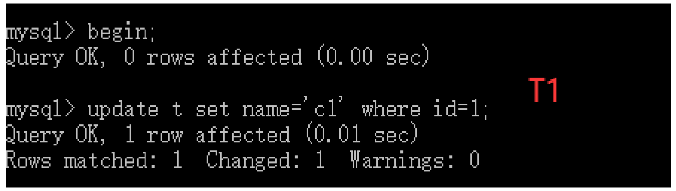
1



```
mysql> select * from information_schema.innodb_trx\G
*************************** 1. row ***************************
                    trx_id: 294155
                 trx_state: LOCK WAIT
               trx_started: 2020-11-08 10:57:36
     trx_requested_lock_id: 2318677534496:354:4:2:2318640615912
          trx_wait_started: 2020-11-08 10:57:36
                trx_weight: 2
       trx_mysql_thread_id: 17
                 trx_query: update t set name='c2' where id=1
       trx_operation_state: starting index read
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 1
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 0
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
       trx_schedule_weight: 1
*************************** 2. row ***************************
                    trx_id: 294154
                 trx_state: RUNNING
               trx_started: 2020-11-08 10:57:31
     trx_requested_lock_id: NULL
          trx_wait_started: NULL
                trx_weight: 3
       trx_mysql_thread_id: 16
                 trx_query: NULL
       trx_operation_state: NULL
         trx_tables_in_use: 0
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 3
         trx_rows_modified: 1
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 0
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
       trx_schedule_weight: NULL
2 rows in set (0.00 sec)
```

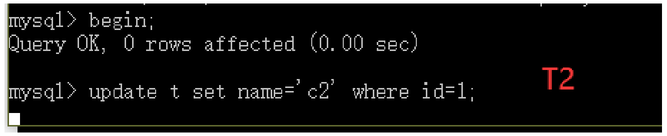
T2: 正在进行锁等待

T1：update结束，还未提交



```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> update t set name='c1' where id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

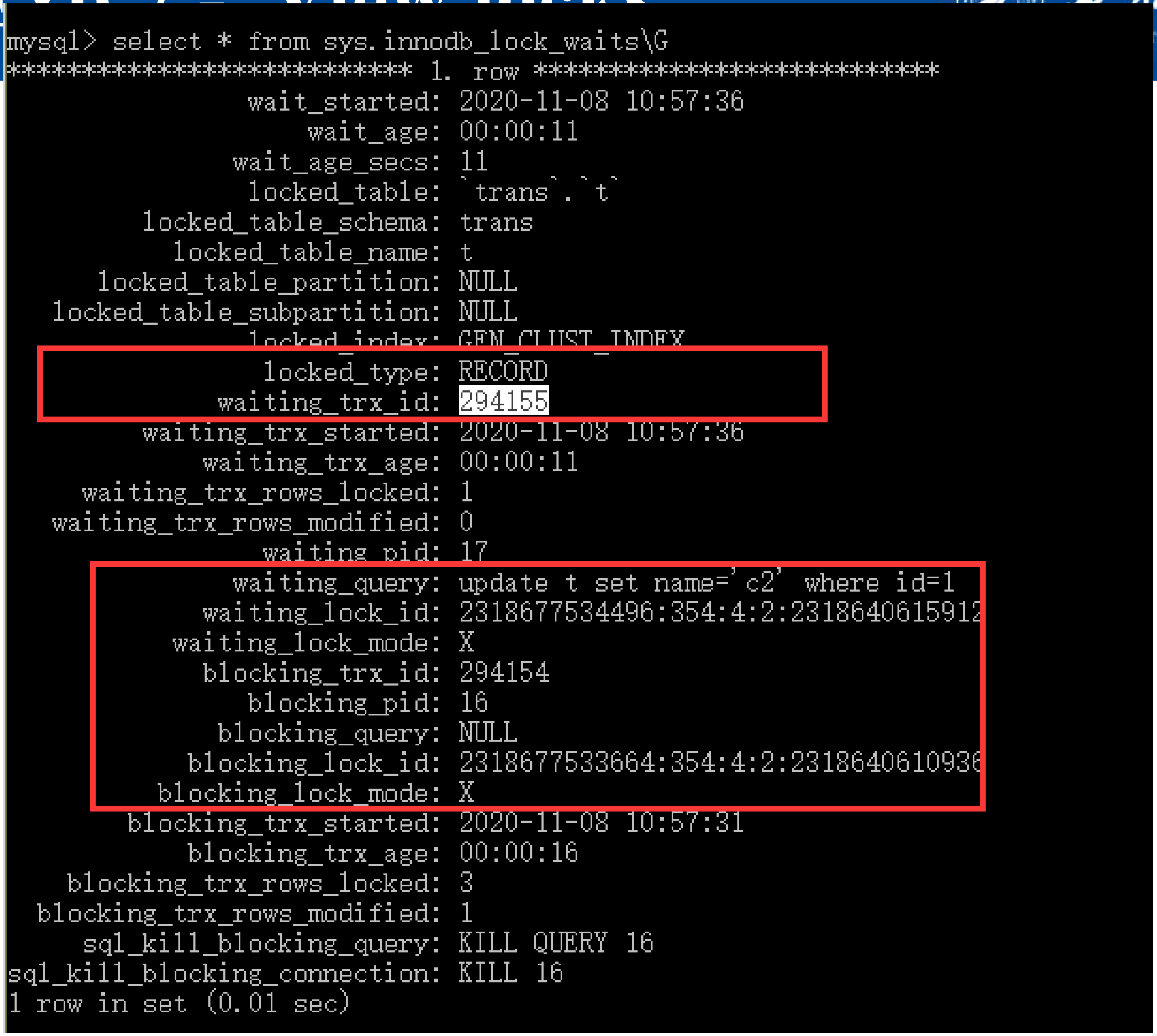
T1



```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> update t set name='c2' where id=1;
```

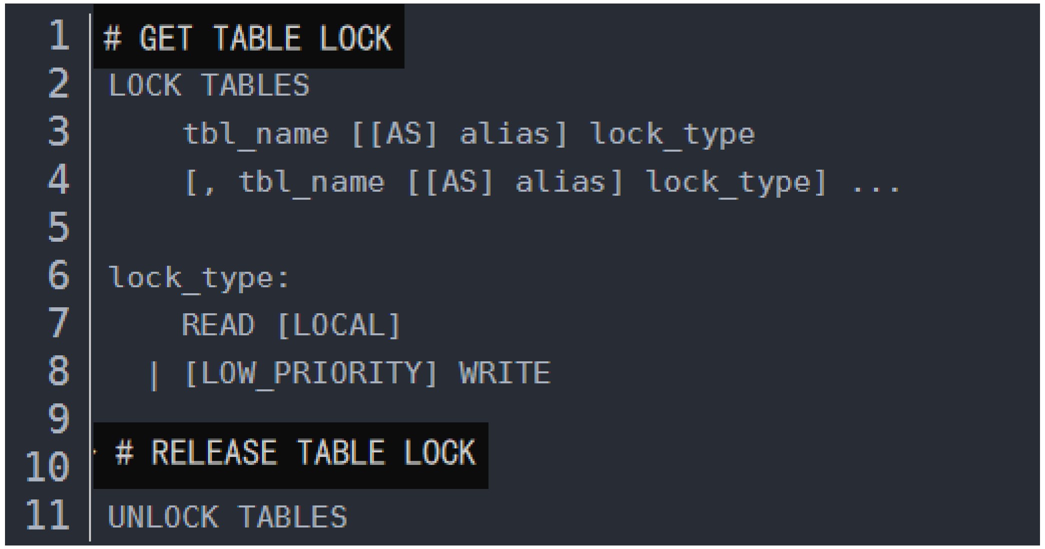
T2

1. Exp 7 – View locks

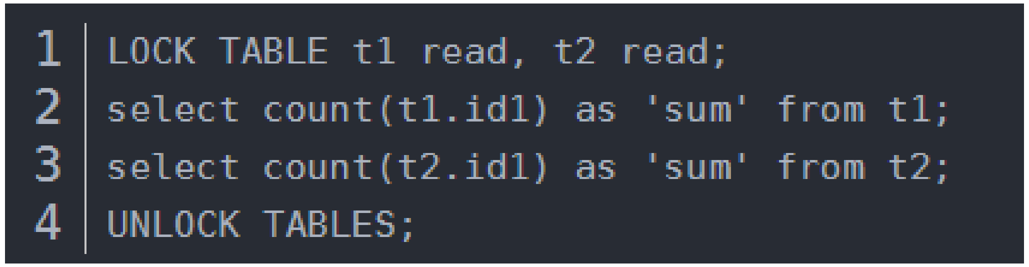


```
mysql> select * from sys.innodb_lock_waits\G
*************************** 1. row ***************************
                wait_started: 2020-11-08 10:57:36
                    wait_age: 00:00:11
               wait_age_secs: 11
                locked_table: `trans`.`t`
         locked_table_schema: trans
           locked_table_name: t
      locked_table_partition: NULL
   locked_table_subpartition: NULL
                locked_index: GEN_CLUST_INDEX
                 locked_type: RECORD
              waiting_trx_id: 294155
         waiting_trx_started: 2020-11-08 10:57:36
             waiting_trx_age: 00:00:11
     waiting_trx_rows_locked: 1
   waiting_trx_rows_modified: 0
                 waiting_pid: 17
               waiting_query: update t set name='c2' where id=1
             waiting_lock_id: 2318677534496:354:4:2:2318640615912
           waiting_lock_mode: X
             blocking_trx_id: 294154
                blocking_pid: 16
              blocking_query: NULL
            blocking_lock_id: 2318677533664:354:4:2:2318640610936
          blocking_lock_mode: X
        blocking_trx_started: 2020-11-08 10:57:31
            blocking_trx_age: 00:00:16
    blocking_trx_rows_locked: 3
  blocking_trx_rows_modified: 1
     sql_kill_blocking_query: KILL QUERY 16
sql_kill_blocking_connection: KILL 16
1 row in set (0.01 sec)
```

1. Exp 7 - Table lock



```
# GET TABLE LOCK
LOCK TABLES
    tbl_name [[AS] alias] lock_type
    [, tbl_name [[AS] alias] lock_type] ...

lock_type:
    READ [LOCAL]
  | [LOW_PRIORITY] WRITE

# RELEASE TABLE LOCK
UNLOCK TABLES
```



```
LOCK TABLE t1 read, t2 read;
select count(t1.id1) as 'sum' from t1;
select count(t2.id1) as 'sum' from t2;
UNLOCK TABLES;
```

1. Exp 7 - Row locks

Use the following statement to get shared lock and exclusive lock :

- **Shared lock (S)** :

  SELECT * FROM table_name WHERE ... LOCK IN SHARE MODE。

Other sessions can query records in the table and also can lock records in 'share mode'. However, if the current transaction is updatin the locked records, it is likely to cause a deadlock.

- **Exclusive lock (X)**:

  SELECT * FROM table_name WHERE ... FOR UPDATE

Other sessions can query the record, but can not lock it (Neither X nor S) but wait to get lock.

1. Exp 7 - View locks

■ Record lock of InnoDB

- **Record Lock： locks a single record.Record Lock always locks the index records， if there is no index in the table, the primary key is used for locking as index implictly.**
- **Gap Lock： locks a range in the table, does not contain the record itself. A lock on an index record gap, or before the first index record and after the last index record.**
- **Next-Key Lock： Gap Lock+Record Lock， locks a range including the record itself. Opened forward and closed back， e.g (5,10】.**

1. Exp 7 - Isolation Levels

lost update:

isolation level （repeatable read ）

table: card , ini value：

id	value
a	100

T1

START

① select value into @v1 from card where id='a';

@V1
100

T2

START

② select value into @v2 from card where id='a';

@V2
100

③ update card set value = @v2 + 200 where id='a';

id	value
a	300

④ **COMMIT**

⑤ update card set value = @v1 + 100 where id='a';

After Update

id	value
a	200

⑥ **ROLLBACK**

⑦ select value from card where id='a';

id	value
a	300

Modification of T2 is not lost
（1st type of lost update）

Notice: Two transaction must execute in differente sessions.

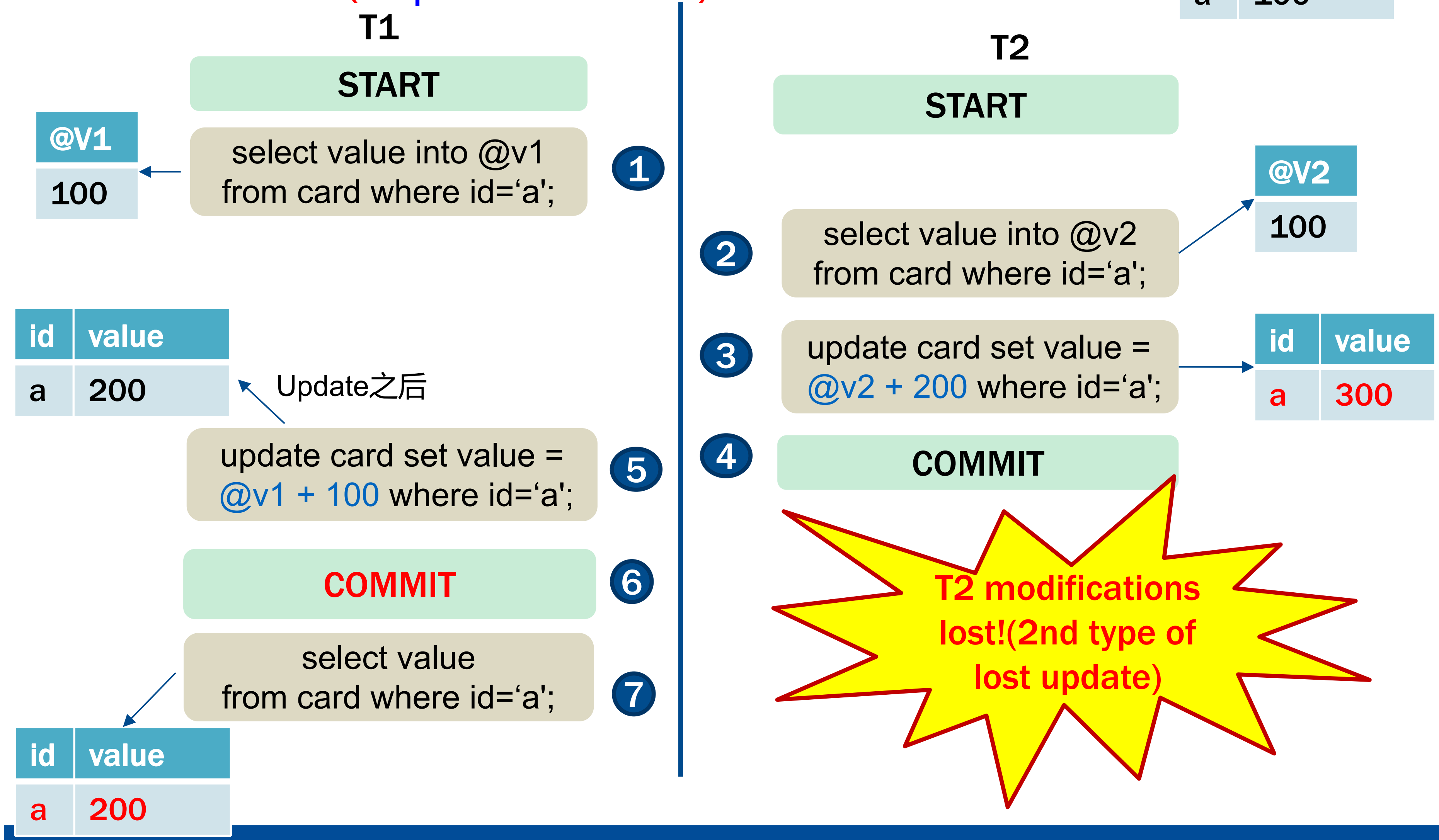
1. Exp 7 - Isolation Levels

lost update:

isolation level （ repeatable read ）

Table: card , ini value：

id	value
a	100



T1

START

① select value into @v1 from card where id=‘a';

@V1
100

T2

START

② select value into @v2 from card where id=‘a';

@V2
100

③ update card set value = @v2 + 200 where id=‘a';

id	value
a	300

④ COMMIT

⑤ update card set value = @v1 + 100 where id=‘a';

Update之后

id	value
a	200

⑥ COMMIT

⑦ select value from card where id=‘a';

id	value
a	200

T2 modifications lost!(2nd type of lost update)

1. Exp 7 - Isolation Levels

西北工業大學 NORTHWESTERN POLYTECHNICAL UNIVERSITY

lost update:

isolation level （read uncommitted）

Table: card , ini value：

id	value
a	100

T1

START

① select value from card where id=‘a';

id	value
a	100

T2

START

② select value from card where id=‘a';

id	value
a	100

③ update card set value = value+200 where id='a';

id	value
a	300

④ **COMMIT**

Update前重读数据value

⑤ update card set value = value+100 where id=‘a';

id	value
a	400

⑥ **COMMIT**

⑦ select value from card where id=‘a';

id	value
a	400

No lost update in this occasion!
(note the difference from the previous page)

1. Exp 7 - Isolation Levels

lost update: isolation level （serializable）

table: card , ini value：

id	value
a	100

T1

START

1. select value into @v1 from card where id=‘a’;

@V1
100

T2

START

2. select value into @v2 from card where id=‘a’;

@V2
100

3. update card set value = @v2 + 200 where id=‘a’;

Block, waiting for write lock

4. update card set value = @v1 + 100 where id=‘a’;

Deadlocks may occur, and system forces T1 lock to be released, execute T2's update not T1's.

5. update card set value = @v2 + 200 where id=‘a’;

id	value
a	300

6. COMMIT

id	value
a	300

7. COMMIT

T2 modifications are not lost! Fixed the problem of lost update!

1. Exp 7 - Isolation Levels

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

Dirty read: isolation level
（ read uncommitted ）

Table: card , ini value：

id	value
a	100

T1

START

id	value
a	100

select value
from card where id=‘a'; 1

id	value
a	300

read uncommitted data

select value
from card where id=‘a'; 4

COMMIT 6

T2

START

2 select value
from card where id=‘a';

id	value
a	100

3 update card set value =
value+200 where id='a';

id	value
a	300

5 **COMMIT**

In T1， it read data updated by other transaction but uncommitted.

1. Exp 7 - Isolation Levels

unrepeatable: isolation level
(read committed)

Table: card , ini value：

id	value
a	100

T1

START

id	value
a	100

1 select value
from card where id=‘a';

id	value
a	100

read the ini value

4 select value
from card where id=‘a';

id	value
a	300

read the committed data

6 select value
from card where id=‘a';

7 COMMIT

T2

START

2 select value
from card where id=‘a';

id	value
a	100

3 update card set value =
value+200 where id=‘a';

id	value
a	300

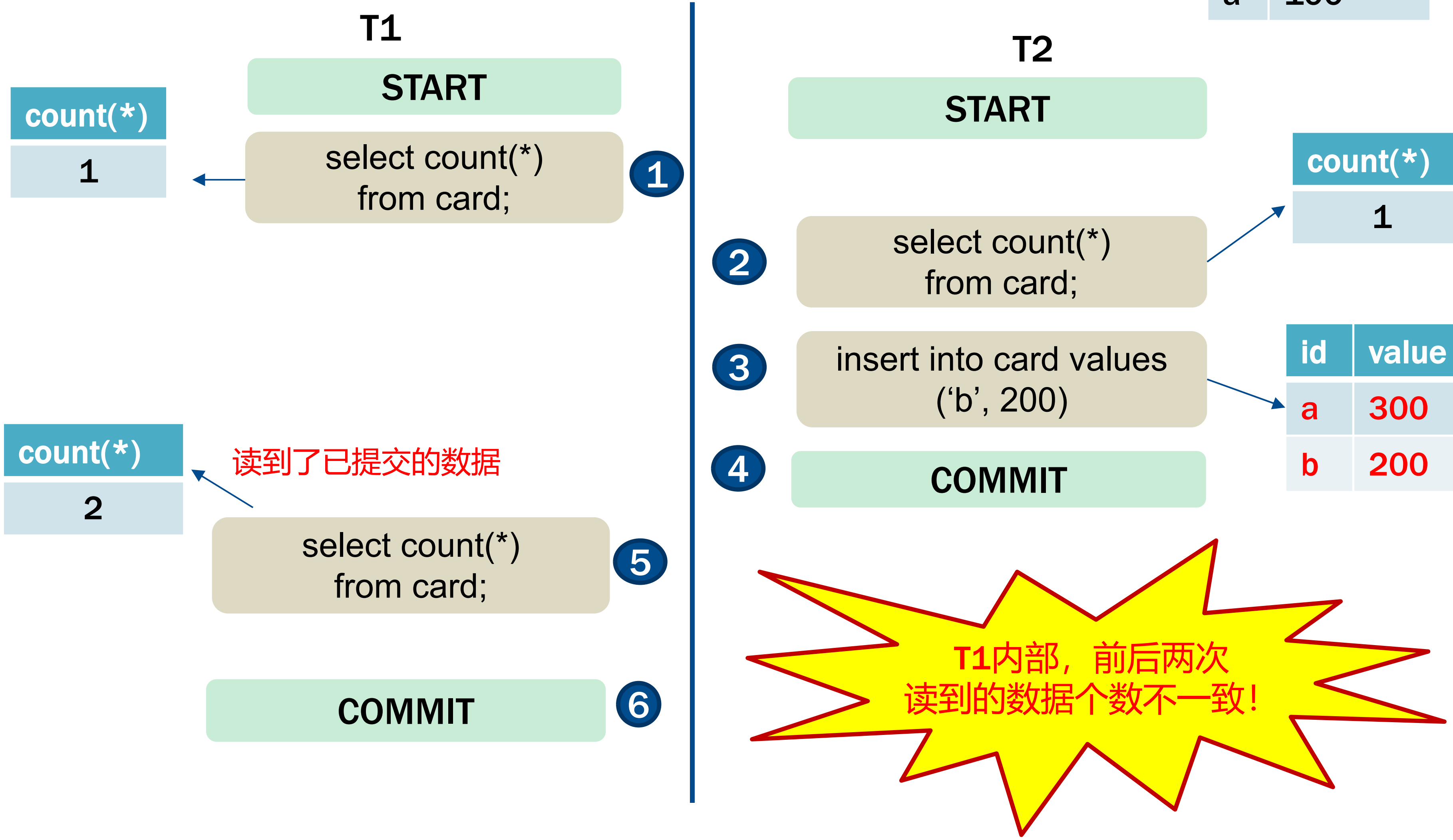
5 COMMIT

Inside T1, the data read before and after are inconsistent!

1. Exp 7 - Isolation Levels

幻读（插入）：隔离级别（ read committed ） 表: card , 初值:

id	value
a	100



T1

START

① select count(*) from card;

count(*)
1

T2

START

② select count(*) from card;

count(*)
1

③ insert into card values ('b', 200)

id	value
a	300
b	200

④ COMMIT

⑤ select count(*) from card;

读到了已提交的数据

count(*)
2

⑥ COMMIT

T1内部，前后两次
读到的数据个数不一致!

1. Exp 7 - Deadlock detect

Under the MySQL default setting, deadlock detection will be performed automatically. After a deadlock is detected, wait for a timeout, and then force the deadlock to end,so that another transaction can continue.

Deadlock found when trying to get lock; try restarting transaction

1. Exp 7 - Transaction log

Binlog file: log file used for transaction recovery

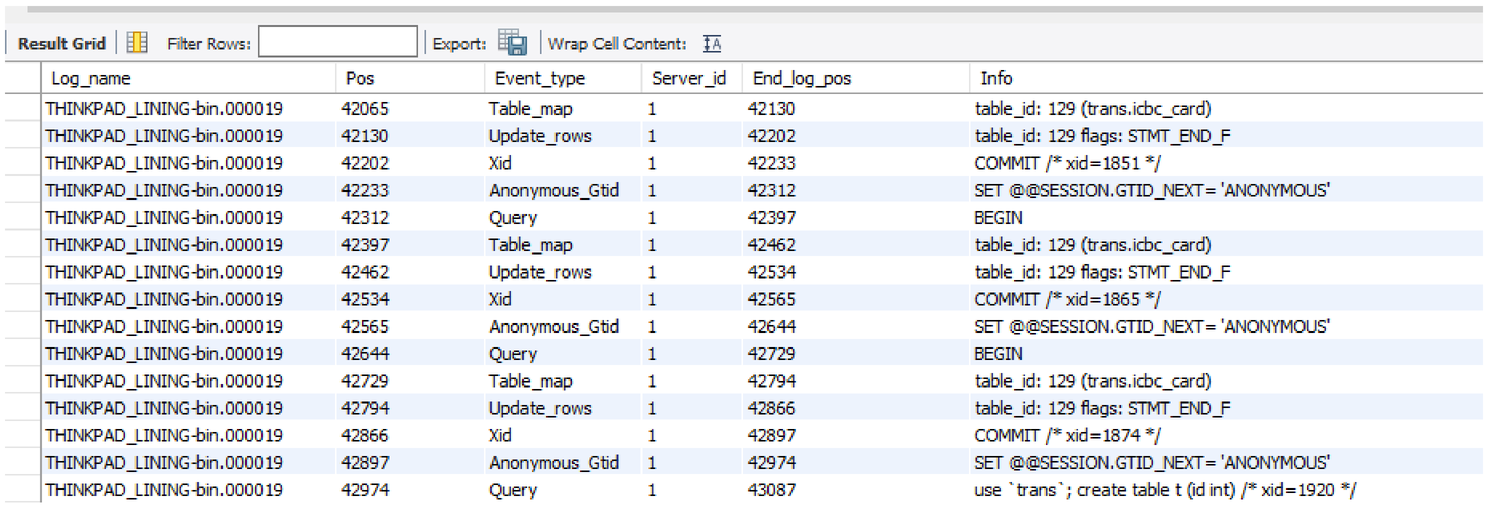
```
-- Displays basic information about binlog
show variables like 'log_bin'; -- to confirm if binlog is on
show master logs;  -- display all binlog files
show master status;  -- Displays the last position of the latest binlog
show binlog events;         -- display all binlog

-- Displays the specified binlog
show binlog events in 'THINKPAD_LINING-bin.000019’   ;

-- Displays the log after a specified location in the binlog.
show binlog events in 'THINKPAD_LINING-bin.000019' from 40080; show
binlog events in 'THINKPAD_LINING-bin.000019' from 40080 limit 10;

--refresh binlog, generate a new binlog file
flush logs;
```

1. Exp 7 - Transaction log

NORTHWESTERN POLYTECHNICAL UNIVERSITY

Binlog file: log file used for transaction recovery

- Row:No SQL statement context information is recorded, only record the modified record.
- Statement: Each SQL that modifies data is recorded in the binlog.
- Mixedlevel: A mixture of the above two. For general statement modification, binlog is stored in the statement format, such as some functions. If the statement cannot perform master-slave replication, binlog is stored in row format. Mysql treats the record differently based on each executed SQL statement.



Log_name	Pos	Event_type	Server_id	End_log_pos	Info
THINKPAD_LINING-bin.000019	42065	Table_map	1	42130	table_id: 129 (trans.icbc_card)
THINKPAD_LINING-bin.000019	42130	Update_rows	1	42202	table_id: 129 flags: STMT_END_F
THINKPAD_LINING-bin.000019	42202	Xid	1	42233	COMMIT /* xid=1851 */
THINKPAD_LINING-bin.000019	42233	Anonymous_Gtid	1	42312	SET @@SESSION.GTID_NEXT= 'ANONYMOUS'
THINKPAD_LINING-bin.000019	42312	Query	1	42397	BEGIN
THINKPAD_LINING-bin.000019	42397	Table_map	1	42462	table_id: 129 (trans.icbc_card)
THINKPAD_LINING-bin.000019	42462	Update_rows	1	42534	table_id: 129 flags: STMT_END_F
THINKPAD_LINING-bin.000019	42534	Xid	1	42565	COMMIT /* xid=1865 */
THINKPAD_LINING-bin.000019	42565	Anonymous_Gtid	1	42644	SET @@SESSION.GTID_NEXT= 'ANONYMOUS'
THINKPAD_LINING-bin.000019	42644	Query	1	42729	BEGIN
THINKPAD_LINING-bin.000019	42729	Table_map	1	42794	table_id: 129 (trans.icbc_card)
THINKPAD_LINING-bin.000019	42794	Update_rows	1	42866	table_id: 129 flags: STMT_END_F
THINKPAD_LINING-bin.000019	42866	Xid	1	42897	COMMIT /* xid=1874 */
THINKPAD_LINING-bin.000019	42897	Anonymous_Gtid	1	42974	SET @@SESSION.GTID_NEXT= 'ANONYMOUS'
THINKPAD_LINING-bin.000019	42974	Query	1	43087	use `trans`; create table t (id int) /* xid=1920 */

1. Exp 7 - Transaction log

Binlog file：binlog for transaction recovery

```
flush logs;
show master status;
Assume：newest：mysql-bin. 000022
```

new created binlog file

↓

Perform regular SQL data operations（including create, insert, update delete operations)

↓

Retrieve the SQL statements before recovery from the log，export as test000022.sql

1. mysqlbinlog.exe mysql-bin.000022 > test_000022.txt
2. Find the location of the log to be recovered (eg.drop TABLE) in the TXT log (at 2413 for this statement).
3. Export the SQL statements before 'DROP TABLE' in the binlog log.

```
mysqlbinlog mysql-bin.000022 -d db1 --skip-gtids --stop-position=2413 > test000022.sql
```

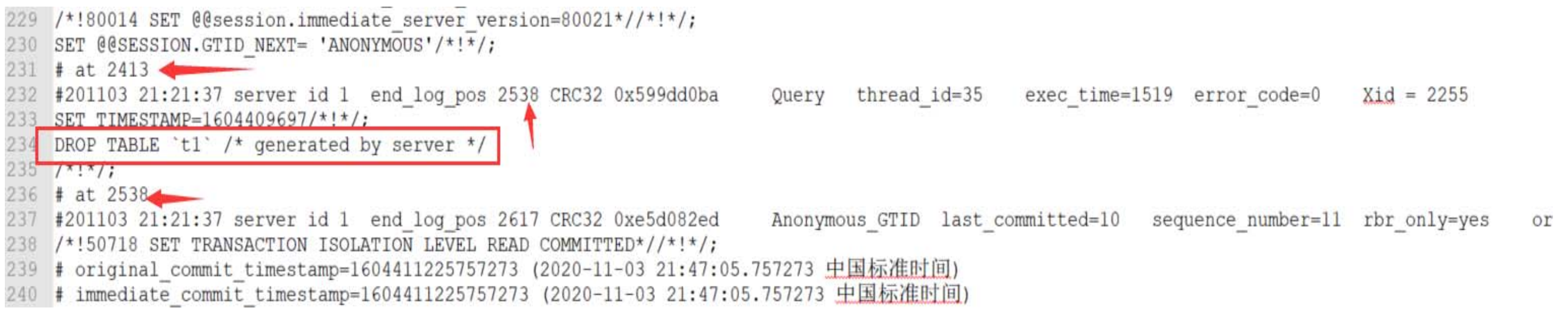
↓

execute the sql file in mysql

```
source C:\ProgramData\MySQL\MySQL Server 8.0\Data\test000022.sql
```

1. Exp 7 - Transaction log

Binlog file：(output file is a txt file)



```
/*!80014 SET @@session.immediate_server_version=80021*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
# at 2413
#201103 21:21:37 server id 1  end_log_pos 2538 CRC32 0x599dd0ba     Query   thread_id=35    exec_time=1519  error_code=0    Xid = 2255
SET TIMESTAMP=1604409697/*!*/;
DROP TABLE `t1` /* generated by server */
/*!*/;
# at 2538
#201103 21:21:37 server id 1  end_log_pos 2617 CRC32 0xe5d082ed     Anonymous_GTID  last_committed=10   sequence_number=11  rbr_only=yes    or
/*!50718 SET TRANSACTION ISOLATION LEVEL READ COMMITTED*//*!*/;
# original_commit_timestamp=1604411225757273 (2020-11-03 21:47:05.757273 中国标准时间)
# immediate_commit_timestamp=1604411225757273 (2020-11-03 21:47:05.757273 中国标准时间)
```

Default directory in Windows：
C:\ProgramData\MySQL\MySQL Server 8.0\Data

2. Exp 8 - Project

You can design and implement a project by yourself or by a group.

10.3.1. Library Management System
10.3.2. Student Status Management System
10.3.3. Ticket Selling Management System
10.3.4. Enterprise personnel management system
10.3.5. Telephone payment management system

in the textbook, exercise in Chapter 9 provides many reference topic for you. Choose one to finish your project .