2018380039\_Dikshya Kafle Experiment 7: DBMS

# Experiment 7：Transaction and Concurrency Control

## Goal

To practice the

1. To grasp the concept transaction, and how to create the transaction.
2. Understand the data inconsistency problems in concurrency operation, and can use lock and ioslation mechanisms

## Content

Suppose that the university allows students to bind the bank card with the campus card. There are the following basic tables in the student database, in which the campus card number (cardid) is the student number:

**icbc\_card(studcardid, icbcid, balance) //campus ID，icbc bank ID，balance of bank card**

**campus\_card(studcardid, balance) // campus card ID，c\_card balance**

**Some example data in this experiment**

**create table icbc\_card(**

**icbcid int,**

**sno varchar(8),**

**balance int**

**);**

**create table campus\_card(**

**sno varchar(8),**

**balance int**

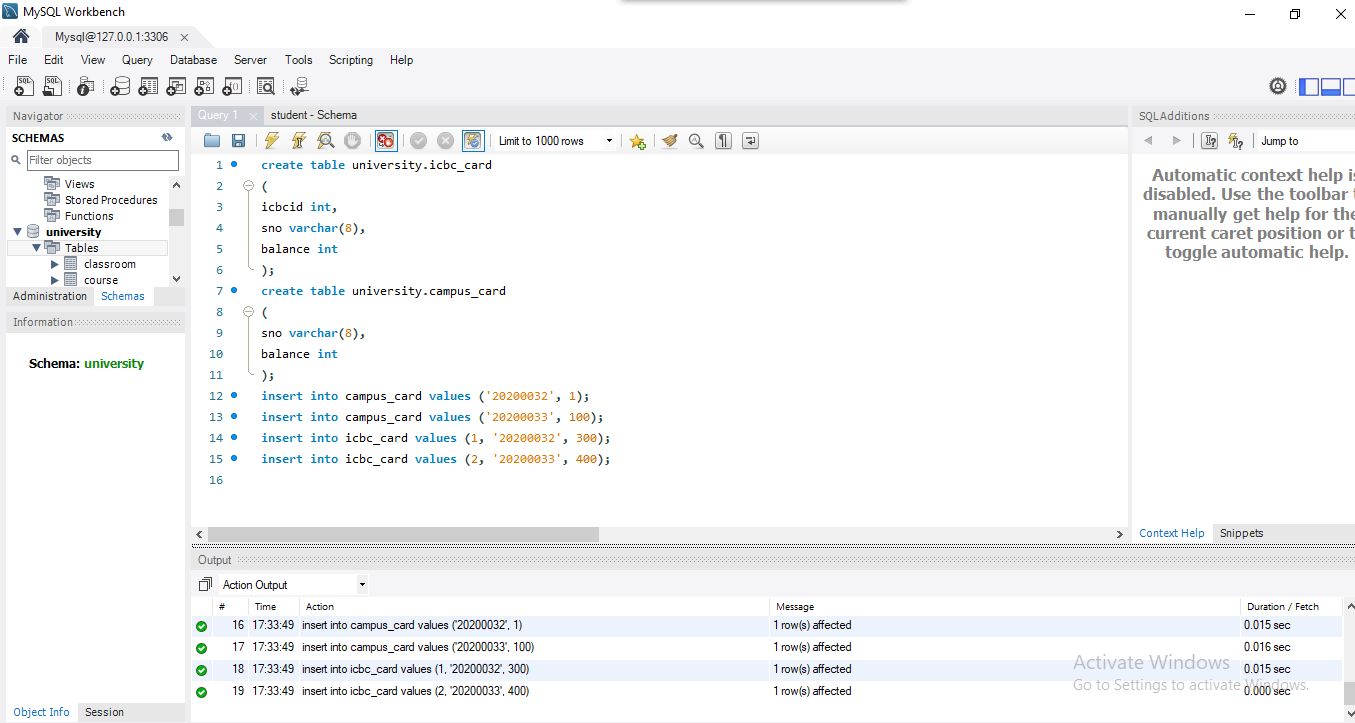
**);**

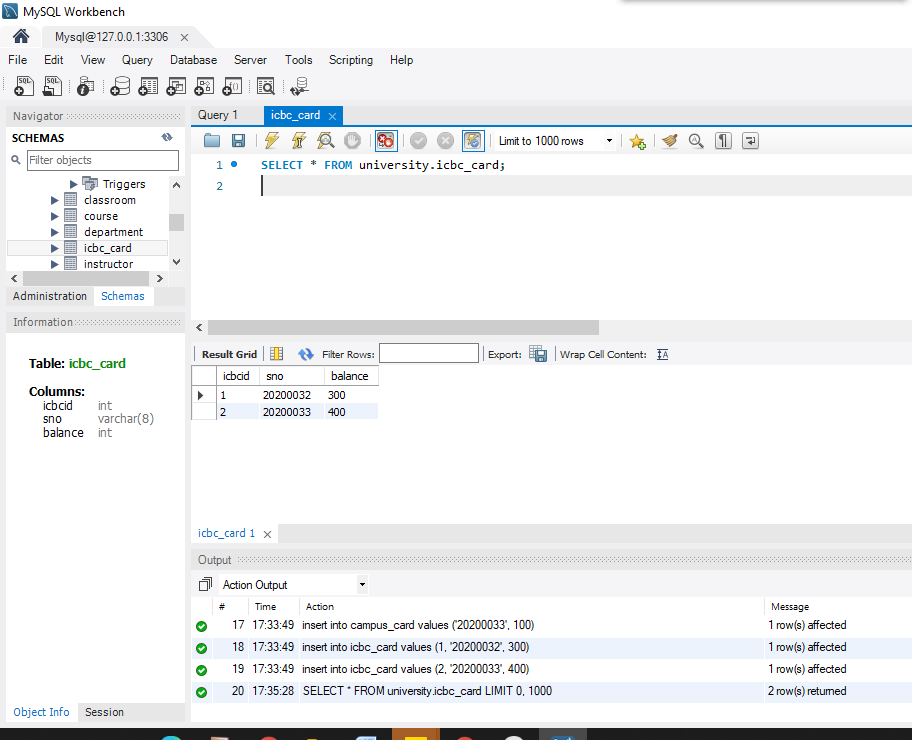
**insert into campus\_card values ('20200032', 1);**

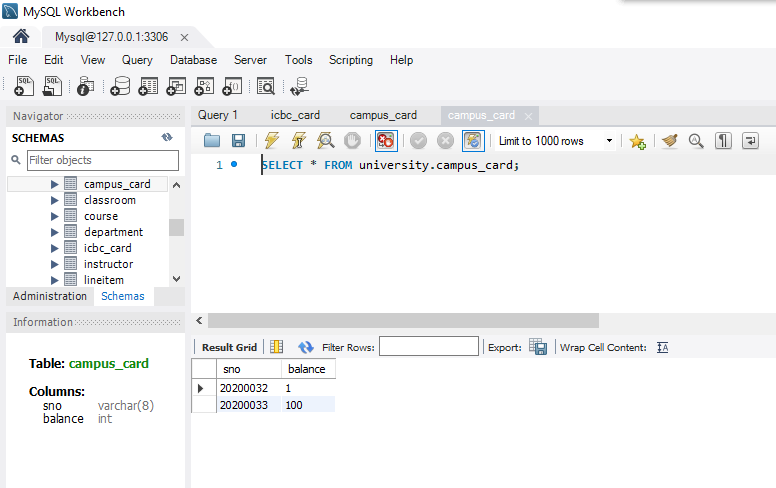
**insert into campus\_card values ('20200033', 100);**

**insert into icbc\_card values (1, '20200032', 300);**

**insert into icbc\_card values (2, '20200033', 400);**

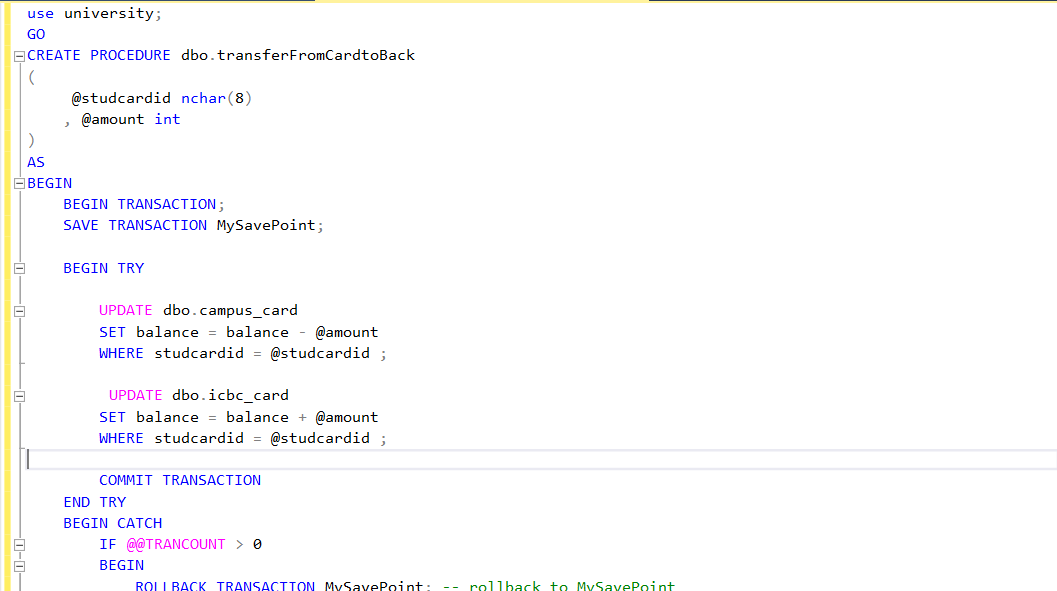


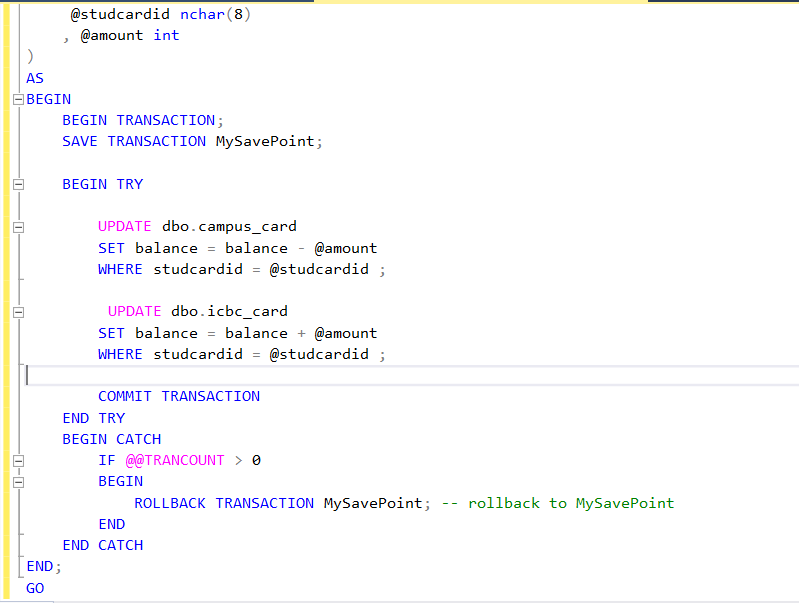




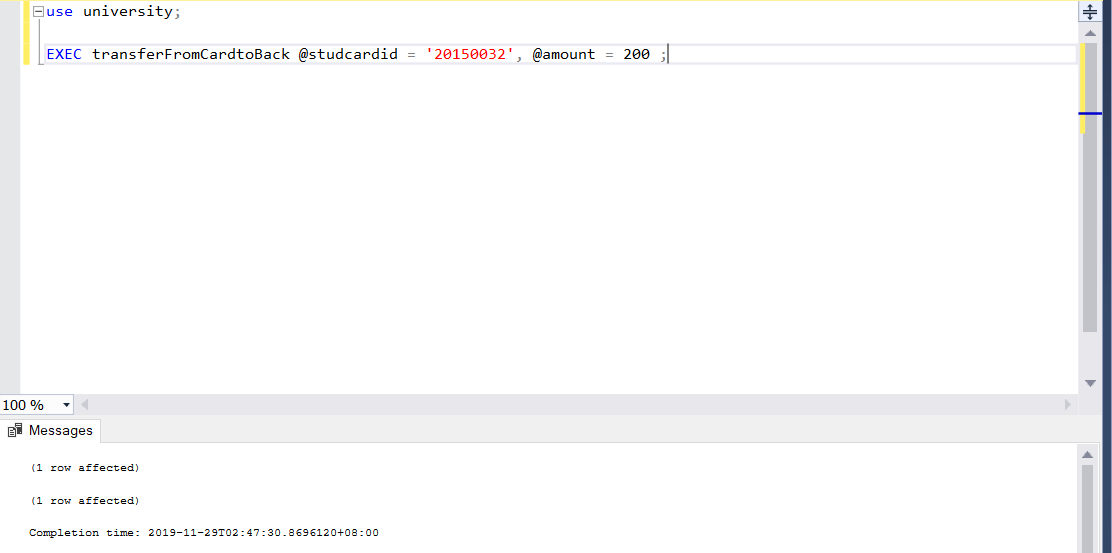
According to the requirements, try to complete the following experiments based on the database above:

1. Write a transaction to achieve the following operations: a student( student number is 20200032) transfers 200 yuan from the bank card to the campus card, and if there is a failure during the transfer process, it will be rolled back.





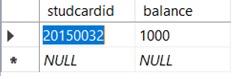
**Executing the procedures:**



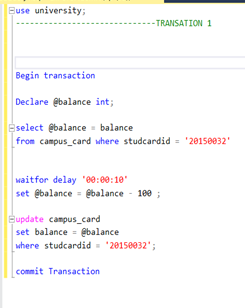
1. According to the database and tables, use specific examples to show several data inconsistency problems: such as missing and modifying, reading dirty data, non repeatable reading and phantom reading (deletion and insertion). If there is any situation that cannot be displayed, please explain the reasons.

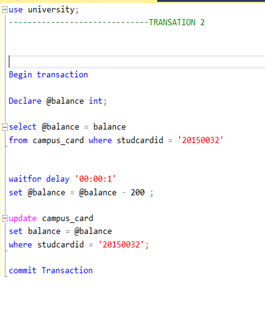
**update lost**

We us the last campus\_card table. We run two transaction T1 and T2 to update the balance attribute for the student with the id 20150032. T1 subtracts 100 from the balance and T2 subtracts 200 from the balance (say the student is buying something using its campus card). T1 waits for 10 seconds and T2 waits only 1 second.

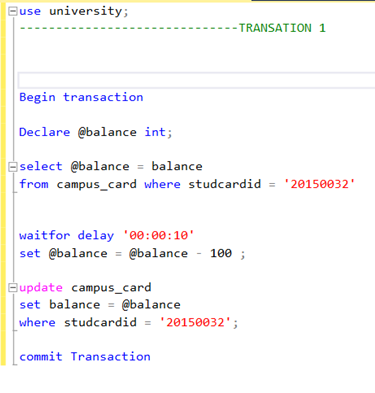


The two transactions (Opening two connections to the database university):

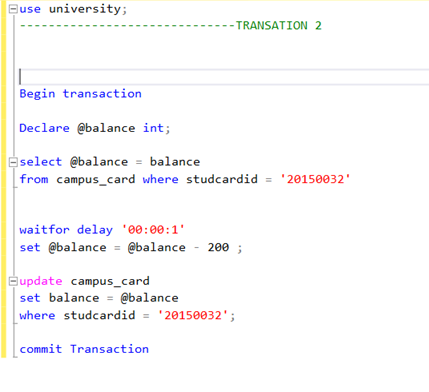




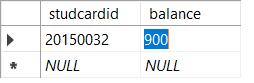
TI:



T2:



We run both transactions at almost the same time and we see the results:

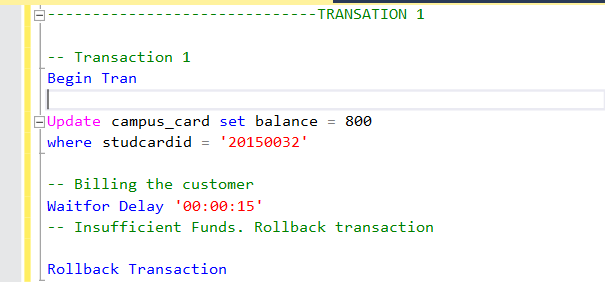


We see how the update of T2 was lost !

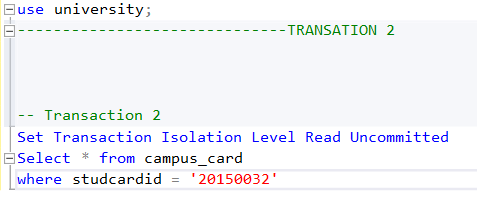
* + 1. **Dirty read**

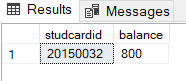
T2 reads data edited by T1 while T1 did not commit yet. In sql server the default isolation level is read committed which solves this problem so that why we change the isolation level for transaction T2.

T1:

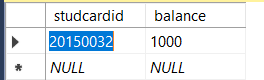


T2 and the result it shows:





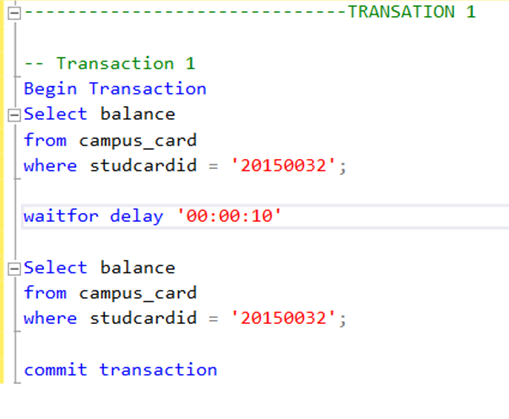
Result of execution (Table After T1 rollback):



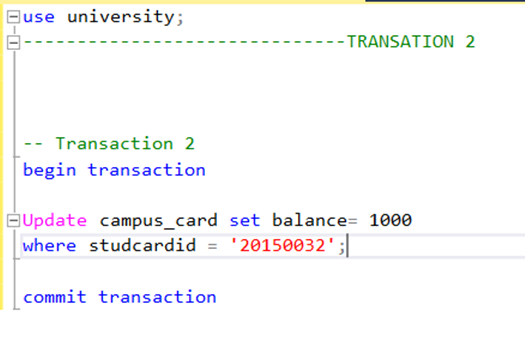
**Unrepeatable read**

In this problem, transaction 1 reads the balance attribute of the student id 20150032 for the first time and wait for 10 seconds. While waiting a transaction 2 updates balance to 1000. Transaction 1 reads the balance again and finds that its values changed and Hence could not get the same read result. This is the Unrepeatable read problem.

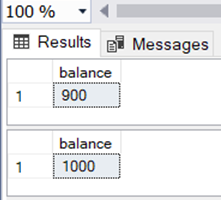
Transaction 1:



Transaction 2:



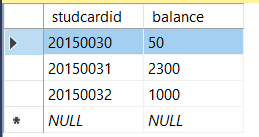
Result:



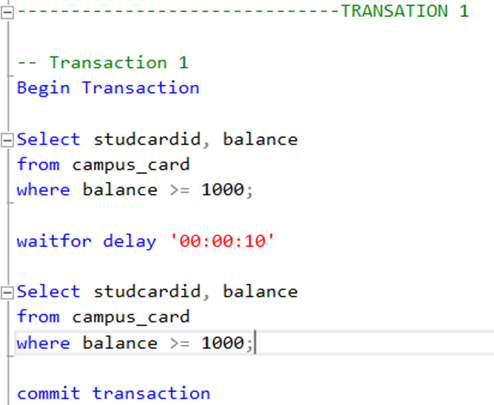
* + 1. **Phantom read**

Transaction T1 executes the same query twice but gets different number of rows in each execution cause T2 inserted a new row while T1 is doing some work.

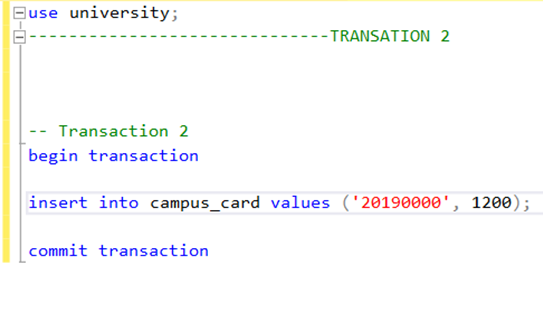
Initial table:



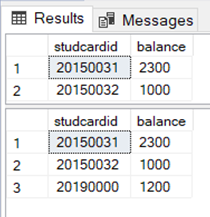
T1:



T2:



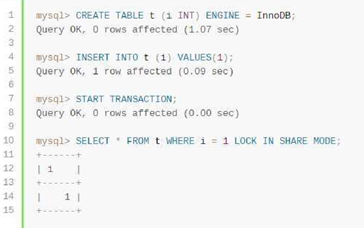
Result:



1. By using the isolation levels or lock mechanism of the database, design solutions to solve the data inconsistency problems you have set in question 2. 
2. Construct two transactions and update one tuple in the database at the same time. Try to use the following SQL commands to view and understand the feed back information of transaction and lock status in the current system.

* show engine innodb status (MySQL 8.0 or 5.7)
* select \* from information\_schema.innodb\_trx (MySQL 8.0 or 5.7)
* select \* from performance\_schema.data\_locks; (MySQL 8.0)
* select \* from sys.innodb\_lock\_waits; (MySQL 8.0)
* select \* from information\_schema.innodb\_lock\_waits (MySQL 5.7)
* select \* from information\_schema.innodb\_locks (MySQL 5.7)

1. Construct a deadlock situation.



1. Construct the transaction containing some ‘savepoint’ and roll back to a savepoint at a certain time. (10 points)
2. Through experiments try to check all kinds of logs in MySQL: query log, error log and slow query log.



1. Use mysqlbinlog to view the transaction log of the database, and try to recover the data according to the following scenarios.

Steps：

1) crate db1, and create table t1, t2 . Table t1,t2 structures are the same: create table t1 (id int);

2) Insert data into t1: 11,12,13

3) Insert data into t2: 21, 22, 23

3) drop table t1;

4) t2 can also be used normally, insert data 24 into t2

After finished the 4th step above, use mysqlbinlog to recover T1 data.

The SQL of the above four steps is as follows:

create database db1;

create table t1(id int);

create table t2(id int);

insert into t1 values(11);

insert into t1 values(12);

insert into t1 values(13);

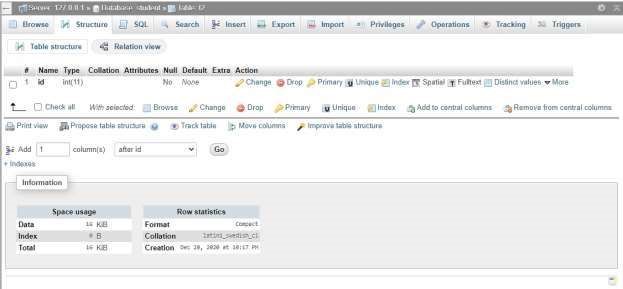
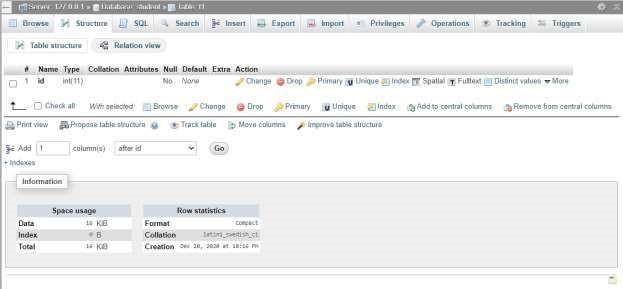
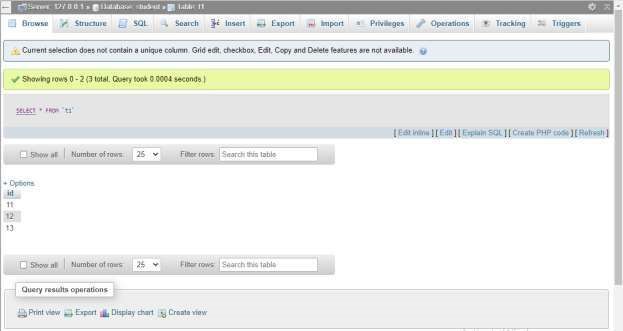
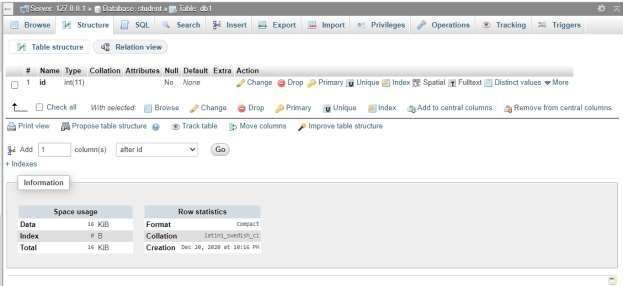
insert into t2 values(21);

insert into t2 values(22);

insert into t2 values(23);

drop table t1;

insert into t2 values(24);



[SELECT \*](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html)FROM `book`

[SELECT \*](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html)FROM `payment`

[SELECT \*](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/select.html)FROM `register` **Problems:**

How to simultaneously execute two transactions. I had a few errors in the design, a couple of connection errors, and a lot of errors in the syntax.

**Solutions:**

Run two MS SQL instances. I look on the internet, in particular YouTube, to get information about these mistakes in order to solve these problems for an answer. I also asked my friends to help me comprehend them. This way I was able to solve the errors.

**Summary:**

In this experiment, we saw how to deal with SQL server transactions. We wrote and executed a real transaction. We discovered the four issues that could occur in a simultaneous manner including dirty read, phantom read, unrepeatable read and update lost while running transaction.

To understand the transaction principle and how the transaction is made. Understand the issues of data inconsistency in the operation of competition, and can use mechanisms of lock and isolation. Suppose the university requires students to attach a campus card to their bank card. In the student database, there are the following simple tables, in which the student number is the campus card number (cardid).

A transaction to accomplish the following operations: a student and changing, reading dirty data, non-repeatable reading and phantom reading (deletion and insertion) using the database's isolation levels or lock mechanism, designing solutions to address the inconsistency of data.