

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №7**

**Дисциплина:** Базы данных

**Тема:** Изучение работы транзакций

Выполнил студент гр. 43501/1

\_\_\_\_\_ Данг Хань  
(подпись)

Руководитель

\_\_\_\_\_ А.В. Мяснов  
(подпись)

“\_\_” \_\_\_\_\_ 2016 г.

Санкт-Петербург

2016

## 1. Цель работы

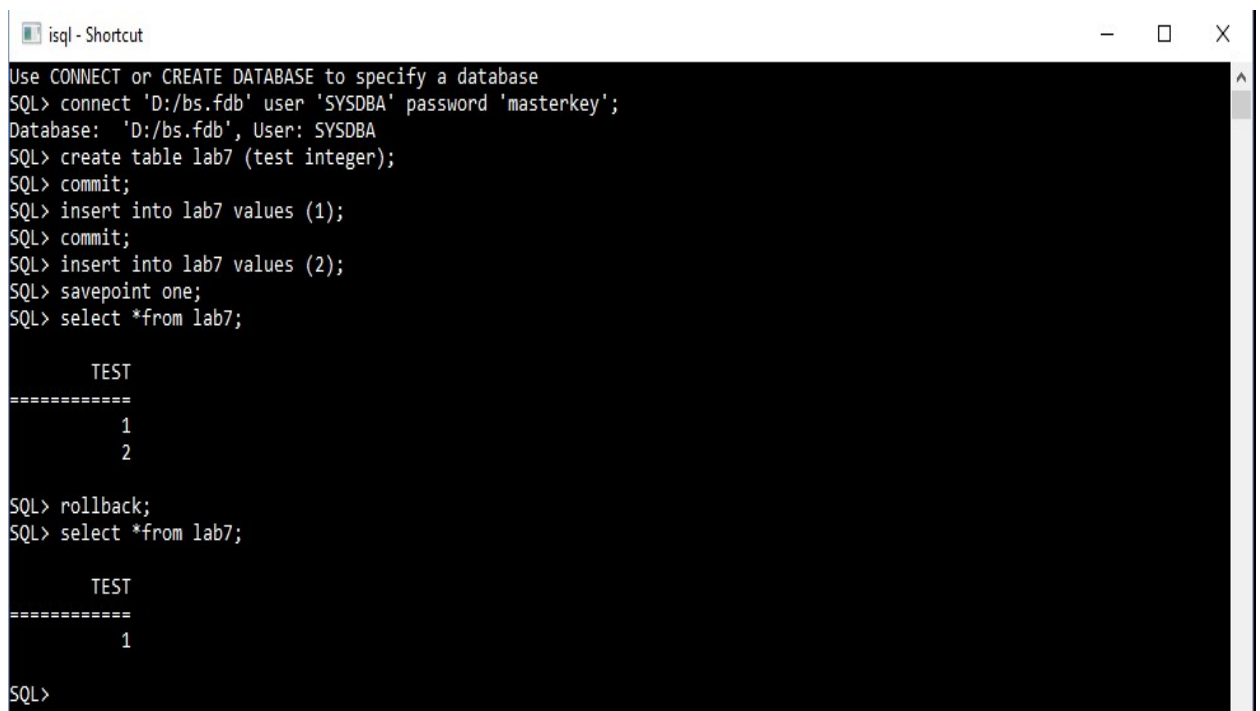
Познакомить студентов с механизмом транзакций, возможностями ручного управления транзакциями, уровнями изоляции транзакций.

## 2. Программа работы

1. Изучить основные принципы работы транзакций.
2. Провести эксперименты по запуску, подтверждению и откату транзакций.
3. Разобраться с уровнями изоляции транзакций в Firebird.
4. Спланировать и провести эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции.
5. Продемонстрировать результаты преподавателю, ответить на контрольные вопросы.

## 3. Выполнение работы

Были проведены эксперименты по запуску, подтверждению и откату транзакций:



```
isql - Shortcut
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect 'D:/bs.fdb' user 'SYSDBA' password 'masterkey';
Database: 'D:/bs.fdb', User: SYSDBA
SQL> create table lab7 (test integer);
SQL> commit;
SQL> insert into lab7 values (1);
SQL> commit;
SQL> insert into lab7 values (2);
SQL> savepoint one;
SQL> select *from lab7;

      TEST
=====
        1
        2

SQL> rollback;
SQL> select *from lab7;

      TEST
=====
        1

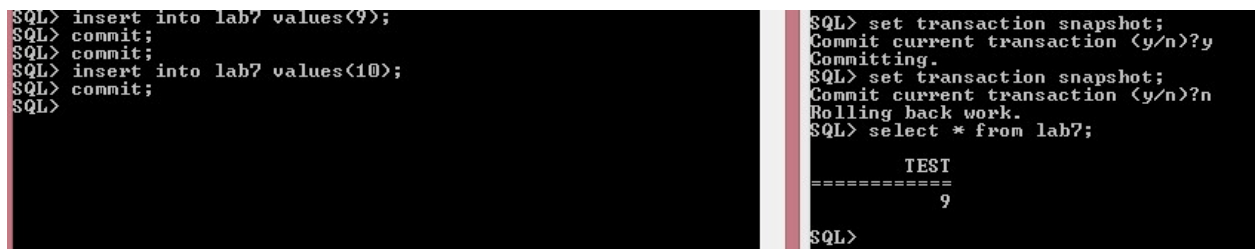
SQL>
```

Была создана таблица Lab7 с одним полем “test”, в это поле были добавлены по очереди два значения, сначала 1, и был выполнен коммит, потом 2, после которого создали точку сохранения. Удалили данные из таблицы Lab7, вернулись к точке сохранения, все данные остались в таблице. Потом вернулись к последнему подтверждению транзакции, в таблице осталось только одно значение 1.

Были проведены эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции:

### Snapshot

Уровень изолированности SNAPSHOT (уровень изолированности по умолчанию) означает, что этой транзакции видны лишь те изменения, фиксация которых произошла не позднее момента старта этой транзакции. Любые подтверждённые изменения, сделанные другими конкурирующими транзакциями, не будут видны в такой транзакции в процессе ее активности без её перезапуска. Чтобы увидеть эти изменения, нужно завершить транзакцию (подтвердить её или выполнить полный откат, но не откат на точку сохранения) и запустить транзакцию заново.



```
SQL> insert into lab7 values(9);
SQL> commit;
SQL> commit;
SQL> insert into lab7 values(10);
SQL> commit;
SQL>

SQL> set transaction snapshot;
Commit current transaction (y/n)?y
Committing.
SQL> set transaction snapshot;
Commit current transaction (y/n)?n
Rolling back work.
SQL> select * from lab7;

      TEST
=====
          9

SQL>
```

Рис.1. Пример

На рисунке 1 видно, что при подключении двух клиентов один вставил в таблицу новое значение (10), но второй клиент не видит изменений.

### Snapshot table stability

Уровень изоляции транзакции SNAPSHOT TABLE STABILITY позволяет, как и в случае SNAPSHOT, также видеть только те изменения, фиксация которых произошла не позднее момента старта этой транзакции. При этом после старта такой транзакции в других клиентских транзакциях невозможно выполнение изменений ни в каких таблицах этой базы данных, уже каким-либо образом измененных первой транзакцией. Все такие попытки в параллельных транзакциях приведут к исключениям базы данных. Просматривать любые данные другие транзакции могут совершенно свободно.

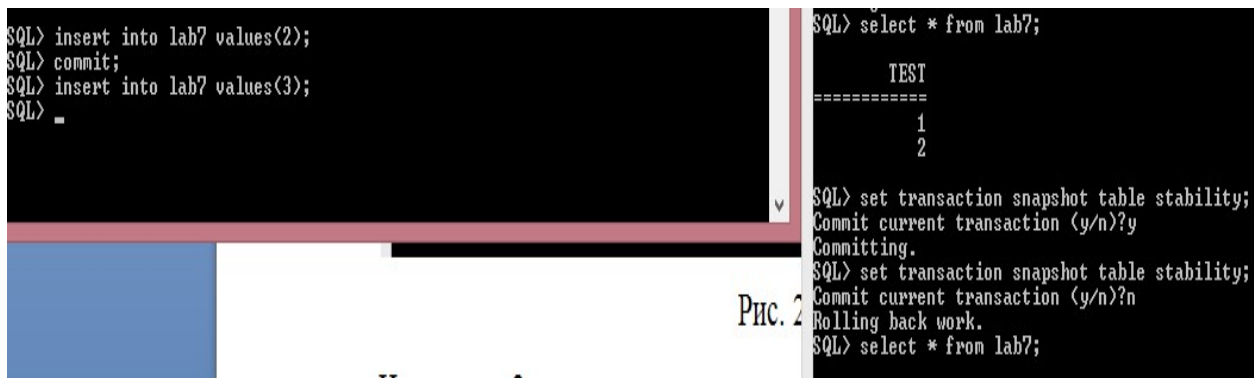
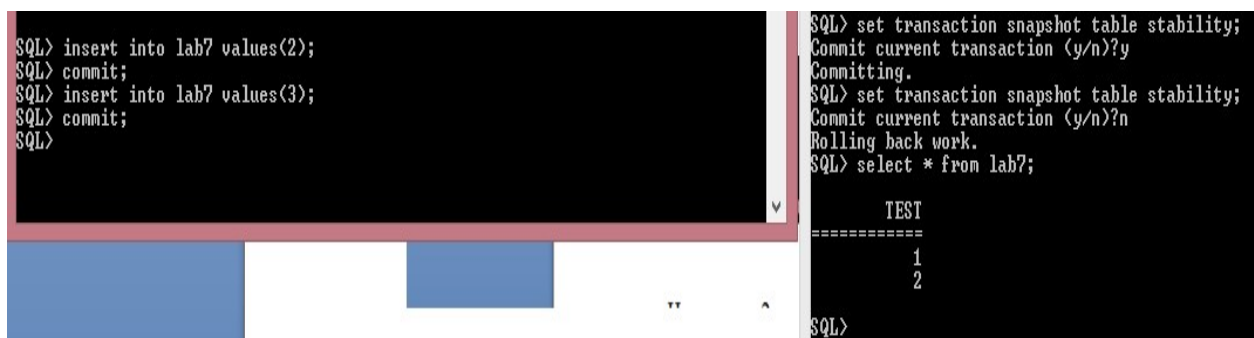


Рис. 2. Пример

На рисунке 2 два клиента подключены к БД: первый добавил данные в таблицу, при эт



ом второй не может завершить процесс вставки.

Рис. 3. Пример

На рисунке 3 видно, что после того, как первый клиент подтвердил транзакцию, второй клиент завершил операцию вставки данных.



Рис. 4. Пример

На рисунке 4 видно, что изменения в таблицах видны только после того, как

оба клиента подтвердили свои транзакции.

### Read committed

Уровень изолированности READ COMMITTED позволяет в транзакции без её перезапуска видеть все подтверждённые изменения данных базы данных, выполненные в других параллельных транзакциях. Неподтверждённые изменения не видны в транзакции и этого уровня изоляции. Для получения обновлённого списка строк интересующей таблицы необходимо лишь повторное выполнение оператора SELECT в рамках активной транзакции READ COMMITTED без её перезапуска.



```
SQL> set transaction read committed;
Commit current transaction (y/n)?n
Rolling back work.
SQL> select * from lab7;

      TEST
=====
         1
         2
         3
         4

SQL> select * from lab7;

      TEST
=====
         1
         2
         3
         4
         5

SQL>

SQL> insert into lab7 values(4);
SQL> commit;
SQL> insert into lab7 values(5);
SQL> commit;
SQL>
```

Рис. 5. Пример

На рисунке 5 видно, что один клиент выполняет вставку данных в таблицу. Второй клиент видит изменения сразу после подтверждения транзакции первым клиентом.

### Record\_Version

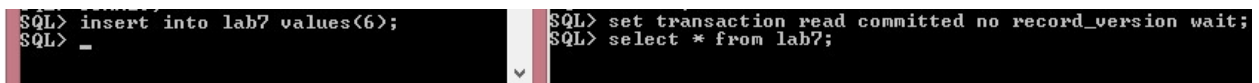
Для этого уровня изолированности можно указать один из двух значений дополнительной характеристики в зависимости от желаемого способа разрешения конфликтов: RECORD\_VERSION и NO RECORD\_VERSION. Как видно из их имён они являются взаимоисключающими.

- NO RECORD\_VERSION (значение по умолчанию) является в некотором роде механизмом двухфазной блокировки. В этом случае транзакция не может прочитать любую запись, которая была изменена параллельной

активной (неподтвержденной) транзакцией.

Если указана стратегия разрешения блокировок NO WAIT, то будет немедленно выдано соответствующее исключение. Если указана стратегия разрешения блокировок WAIT, то это приведёт к ожиданию завершения или откату конкурирующей транзакции. Если конкурирующая транзакция откатывается, или, если она завершается и её идентификатор старше (меньше), чем идентификатор текущей транзакции, то изменения в текущей транзакции допускаются. Если конкурирующая транзакция завершается и её идентификатор новее (больше), чем идентификатор текущей транзакции, то будет выдана ошибка конфликта блокировок.

- При задании RECORD\_VERSION транзакция всегда читает последнюю подтверждённую версию записей таблиц, независимо от того, существуют ли изменённые и ещё не подтверждённые версии этих записей. В этом случае режим разрешения блокировок (WAIT или NO WAIT) никак не влияет на поведение транзакции при её старте.

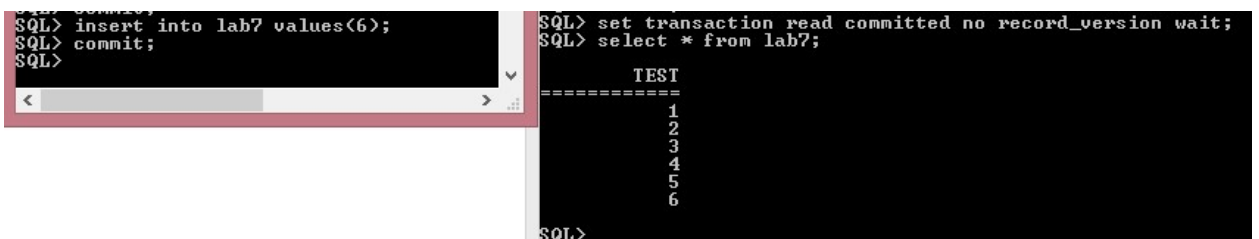


```
SQL> insert into lab7 values(6);
SQL> _

SQL> set transaction read committed no record_version wait;
SQL> select * from lab7;
```

Рис. 6. Пример

Из рисунка 6 видно, что первый клиент добавляет данные в таблицу, второй клиент не может выполнить команду select, пока первый клиент не закончит.



```
SQL> insert into lab7 values(6);
SQL> commit;
SQL>

SQL> set transaction read committed no record_version wait;
SQL> select * from lab7;

TEST
-----
1
2
3
4
5
6
SQL>
```

Рис.7. Пример

После подтверждения первым клиентом транзакции, второй клиент завершает выполнение команды select.



```
SQL> insert into lab7 values(9);
SQL> _

SQL> set transaction read committed no record_version no wait;
SQL> select * from lab7;

      TEST
=====
      1
      2
      3
      4
      5
      6
      7
      8
Statement failed, SQLSTATE = 40001
lock conflict on no wait transaction
-deadlock
SQL>
```

Рис. 8. Пример

На рисунке 8 видно, что первый клиент добавляет данные в таблицу. При обращении вторым клиентом к таблице у него появляется исключение.

#### 4. Вывод

В данной лабораторной работе были изучены основные принципы работы транзакций, уровни изоляции.

Транзакции необходимы для обеспечения целостности БД в соответствии с требованиями предметной области базы данных.

Уровни изоляции транзакций уменьшают возможности параллельной обработки данных в таблицах. При обращении к данным БД используется наиболее оптимальный для данной транзакции уровень изоляции для того, чтобы не потерять в производительности работы базы.