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

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

Дисциплина: «Базы данных»

Тема: «SQL-программирование: Изучение механизма транзакций»

Выполнил студент гр. 43501/3 _____ М.Ю. Попсуйко
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

“___” _____ 2016 г.

Санкт-Петербург

2016

Оглавление

Цель работы:

Познакомиться с механизмом транзакций, возможностями ручного управления транзакциями, уровнями изоляции транзакций.

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.

Выполнение работы:

Основные принципы работы транзакций:

Транзакция – это задача, при которой выполняется одна или несколько операций, каждая из которых должна быть завершена в последовательности. Изменения, выполненные транзакцией, могут быть зафиксированы, если все операции в транзакции завершены. Изменения, внесенные в базу данных невидимы пользователям, пока все операции транзакции не завершаться успешно.

Транзакция начинается автоматически с момента подключения пользователя к СУБД. Завершение происходит если:

1. Подать команду COMMIT, тем самым зафиксировать транзакцию, т.е. записать на диск изменения в БД, которые были сделаны в процессе выполнения транзакции;
2. Подать команду ROLLBACK – выполнить откат транзакции – возврат БД в исходное состояние;
3. Отсоединение пользователя от СУБД, либо сбой системы.

Эксперименты по запуску, подтверждению и откату транзакций:

Создадим тестовую базу данных и подключимся к ней:

```
      SQL> create database
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB'
user 'SYSDBA' password 'masterkey';
      SQL> connect
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB'
user 'SYSDBA' password 'masterkey';
      Database:
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB',
User: SYSDBA
      SQL>
```

Проведем тесты по запуску, подтверждению и откату транзакций:

```
      SQL> create table test_table (id integer not null primary key); --
создаем тестовую таблицу с единственным полем
      SQL> commit;
      SQL> insert into test_table values(1);
      SQL> commit; -- выполним commit
```

```
      SQL> rollback; --видно, что откат не сработал, значение не изменилось
      SQL> select * from test_table;

                ID
      ============
                 1

      SQL> insert into test_table values(2); --добавляем следующую запись, не
выполняя commit
      SQL> select * from test_table;

                ID
      ============
                 1
                 2

      SQL> rollback; --в данном случае откат сработал
      SQL> select * from test_table;

                ID
      ============
                 1

      SQL> insert into test_table values(2);
      SQL> savepoint one;
      -- создадим точку сохранения. Точка сохранения создается для команды
rollback
      SQL> delete from test_table;
      SQL> insert into test_table values(3);
      SQL> select * from test_table;

                ID
      ============
                 3

      SQL> rollback to one;
      SQL> select * from test_table;

                ID
      ============
                 1
                 2
      -- после того, как мы удалили исходные записи и добавили новую, мы
выполнили откат к точке сохранения, в результате мы видим исходные данные
```

Уровни изоляции транзакций в Firebird:

Уровень изолированности транзакций – это значений, определяющее уровень, при котором в транзакции допускаются несогласованные данные, то есть степень изолированности одной транзакции от другой. От уровня изолированности изменяется точность данных и количество параллельно выполняемых транзакций: чем выше уровень, тем точность повышается, количество параллельно выполняемых операций уменьшается, чем ниже, тем точность уменьшается – количество операций увеличивается.

Уровни изоляции транзакций в Firebird:

1. Read committed – самый низкий уровень изоляции. На этом уровне вид состояния базы данных, измененного в процессе выполнения транзакции, изменяется каждый раз, когда подтверждаются версии записей. Этот уровень подходит для операций

реального времени над большим объемом данных, т.к. сокращает количество конфликтов данных. Этот уровень изоляций делает возможным чтения только подтвержденных данных.

2. Snapshot (параллельность) – он изолирует вид БД для транзакции изменениями на уровне строк только для существующих строк. Транзакция гарантирует неизменный вид БД, который до завершения транзакции не будет зависеть от любых подтвержденных изменений в других транзакциях. Все операции чтения данных видят БД на момент запуска транзакции. Такой уровень полезен при оформлении отчетов, которые могут оказаться ошибочными, если не будут выполняться с воспроизводимым видом данных.
3. SNAPSHOT TABLE STABILITY (согласованность) – он гарантирует получение данных в неизменном состоянии, который останется внешне согласованным в пределах базы данных, пока продолжается транзакция. Транзакции чтения/записи не могут читать таблицы, блокированные данным уровнем. Никакая другая транзакция не может изменять строки в используемых таблицах.

Транзакция обладает следующими свойствами:

1. Транзакция может быть выполнена только в том условии, если все операции в ней выполняется, иначе транзакция не выполнится;
2. Все транзакции, желающие обратиться к базе данных физически обрабатываются последовательно, изолированно друг от друга, но пользователь видит это как параллельный процесс;
3. По мере выполнения транзакций данные переходят из одного согласованного состояния в другое – транзакция не разрушает согласованности данных;
4. При успешном завершении транзакции, изменения, совершенные ею, ни при каких условиях не могут быть потеряны.

Эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции:

Read committed:

Терминал 1:

```
      SQL>  connect
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB'
user 'SYSDBA' password 'masterkey';
      Database:
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB',
User: SYSDBA
      SQL> select * from RC_table;

                ID
      ============
                 1
                 2

      SQL> insert into RC_table values(3);
      SQL> commit;
```

Терминал 2:

```
      SQL>  connect
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB'
user 'SYSDBA' password 'masterkey';
```

```
      Database:
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\TEST.FDB',
User: SYSDBA
      SQL> select * from RC_table;

                ID
      ============
                 1
                 2

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

                ID
      ============
                 3
                 1
                 2
```

Эксперимент: я открыл две одинаковые базы данных в двух терминалах. Посмотрел состав одной и той же таблицы. Затем в терминале 2 включил read committed. Создал через первый терминал новую запись в этой таблице. Во втором терминале посмотрел таблицу снова. Результат не был показан до момента, пока я не выполнил commit в первой таблице.

Snapshot:

Терминал 1:

```
      Database:
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\test.fdb',
User: SYSDBA
      SQL> select * from S_table;

                ID
      ============
                 1
                 2

      SQL> insert into S_table values(3);
      SQL> commit;
      SQL> select * from S_table;

                ID
      ============
                 1
                 2
                 3
```

Терминал 2:

```
      SQL> connect
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\test.fdb'
user 'SYSDBA' password 'masterkey';
      Database:
'C:\Users\maxim\OneDrive\Documents\Semester_7\Database\Database\test.fdb',
User: SYSDBA
      SQL> select * from S_table;

                ID
      ============
                 1
```

```
           2

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

          ID
============
           1
           2
```

Во втором терминале включил Snapshot. Добавил в первом терминале запись. Во втором терминале. Сделал выборку всех данных из таблицы. Новой записи не появилось даже после выполнения commit в первом терминале.

Snapshot table stability:

Терминал 1:

```
SQL> select * from SN_table;

          ID
============
           1
           2

SQL> update SN_table set id=10 where id=1;
SQL> select * from SN_table;

          ID
============
          10
           2

SQL> commit;
SQL> update SN_table set id=1 where id=10;
SQL> commit;
SQL> select * from SN_table;

          ID
============
           1
           2
```

Терминал 2:

```
SQL> set transaction isolation level snapshot table stability;
Commit current transaction (y/n)?n
Rolling back work.
SQL> select * from SN_table;

          ID
============
           1
           2

SQL> update SN_table set id=20 where id=1;
Statement failed, SQLSTATE = 40001
Deadlock
- update conflicts with concurrent update
- concurrent transaction number is 64
SQL> update SN_table set id=20 where id=1;
```

```
SQL> select * from SN_table;
          ID
============
          20
           2
```

Достоинства и недостатки механизмов транзакций:

+ Данный механизм обеспечивает целостность базы данных;

+ Обеспечивает правильную работу между несколькими пользователя на одних данных;

+ При сбоях системы, данные работа над которыми осуществлялась транзакцией, остаются не затронуты;

- Возможность возникновения тупиков между транзакциями. Пока одна транзакция не завершилась, другая для работы с этими данными должна ждать;

- Не поддерживают одновременного изменения одного и того блока транзакций (одно теряется);

Выводы:

В результате выполнения данной лабораторной работы познакомился с механизмами транзакций. Ознакомился с возможностями ручного управления транзакциями и уровнями изоляции транзакций.

Транзакции очень полезны, так как позволяют поддерживать целостность данных при работе нескольких клиентов над одними данными. При сбоях, транзакции являются единственными единицами восстановления данных. Но следует также выделить и их недостатки: как говорилось выше, транзакции не поддерживают работу нескольких пользователей над одним и тем же блоком, при параллельной работе над одними данными одна из транзакций должна ожидать, пока другая завершит свою работу.