Санкт-Петербургский государственный политехнический университет
Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №6**

**Дисциплина**: Базы данных

**Изучение работы транзакций**

Выполнила студентка гр. 43501/4 Перетятько Е. В.

Преподаватель: Мяснов А. В.

Санкт-Петербург
2015

## Цели работы

Познакомить студентов 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.

Работа проводится в IBExpert. Для проведения экспериментов параллельно запускается несколько сессий связи с БД, в каждой сессии настраивается уровень изоляции транзакций. Выполняются конкурентные операции чтения/изменения данных в различных сессиях, а том числе приводящие к конфликтам.

**Транзакция** группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Для транзакций существует два режима доступа к данным базы данных: READ WRITE и READ ONLY.

- При режиме доступа READ WRITE операции в контексте данной транзакции могут быть как операциями чтения, так и операциями изменения данных. Это режим по умолчанию.
- В режиме READ ONLY в контексте данной транзакции могут выполняться только операции выборки данных SELECT. Любая попытка изменения данных в контексте такой транзакции приведет к исключениям базы данных. Однако это не относится к глобальным временным таблицам (GTT), которые разрешено модифицировать в READ ONLY транзакциях.

При работе с одной и той же базой данных нескольких клиентских приложений могут возникать блокировки. Блокировки могут возникать, когда одна транзакция вносит неподтвержденные изменения в строку таблицы или удаляет строку, а другая транзакция пытается изменять или удалять эту же строку. Такие блокировки называются конфликтом обновления. Блокировки также могут возникнуть и в других ситуациях при использовании некоторых уровней изоляции транзакций.

Существуют два режима разрешения блокировок: WAIT и NO WAIT.

В режиме WAIT (режим по умолчанию) при появлении конфликта с параллельными транзакциями, выполняющими конкурирующие обновления данных в той же базе данных, такая транзакция будет ожидать завершения конкурирующей транзакции путем ее подтверждения (COMMIT) или отката (ROLLBACK). Иными словами, клиентское приложение будет переведено в режим ожидания до момента разрешения конфликта.

Если установлен режим разрешения блокировок NO WAIT, то при появлении конфликта блокировки данная транзакция немедленно вызовет исключение базы данных.

Уровень изолированности транзакций – значение, определяющее уровень, при котором в транзакции допускаются несогласованные данные, то есть степень изолированности одной транзакции от другой. Изменения, внесенные некоторым оператором, будут видны всем последующим операторам, запущенным в рамках этой же транзакции, независимо от ее уровня изолированности. Изменения произведенные в рамках другой транзакции остаются невидимыми для текущей транзакции до тех пор пока они не подтверждены. Уровень изолированности, а иногда, другие атрибуты, определяют, как

транзакции будут взаимодействовать с другой транзакцией, которая хочет подтвердить изменения.

## 1. Изучить основные принципы работы транзакций

**Транзакция** - это неделимая, с точки зрения воздействия на СУБД, последовательность операций манипулирования данными. Для пользователя транзакция выполняется по принципу "*все или ничего*", т.е. либо транзакция выполняется целиком и переводит базу данных из одного *целостного состояния* в другое *целостное состояние*, либо, если по каким-либо причинам, одно из действий транзакции невыполнимо, или произошло какое-либо нарушение работы системы, база данных возвращается в исходное состояние, которое было до начала транзакции (происходит откат транзакции).

Транзакция обладает **четырьмя важными свойствами**:

- **Атомарность.** Транзакция выполняется как атомарная операция - либо выполняется вся транзакция целиком, либо она целиком не выполняется.
- **Согласованность.** Транзакция переводит базу данных из одного согласованного (целостного) состояния в другое согласованное (целостное) состояние. Внутри транзакции согласованность базы данных может нарушаться.
- **Изоляция.** Транзакции разных пользователей не должны мешать друг другу (например, как если бы они выполнялись строго по очереди).
- **Долговечность.** Если транзакция выполнена, то результаты ее работы должны сохраниться в базе данных, даже если в следующий момент произойдет сбой системы.

Транзакция обычно начинается автоматически с момента присоединения пользователя к СУБД и продолжается до тех пор, **пока не произойдет одно из следующих событий**:

- Подана команда COMMIT WORK (зафиксировать транзакцию).
- Подана команда ROLLBACK WORK (откатить транзакцию).
- Произошло отсоединение пользователя от СУБД.
- Произошел сбой системы.

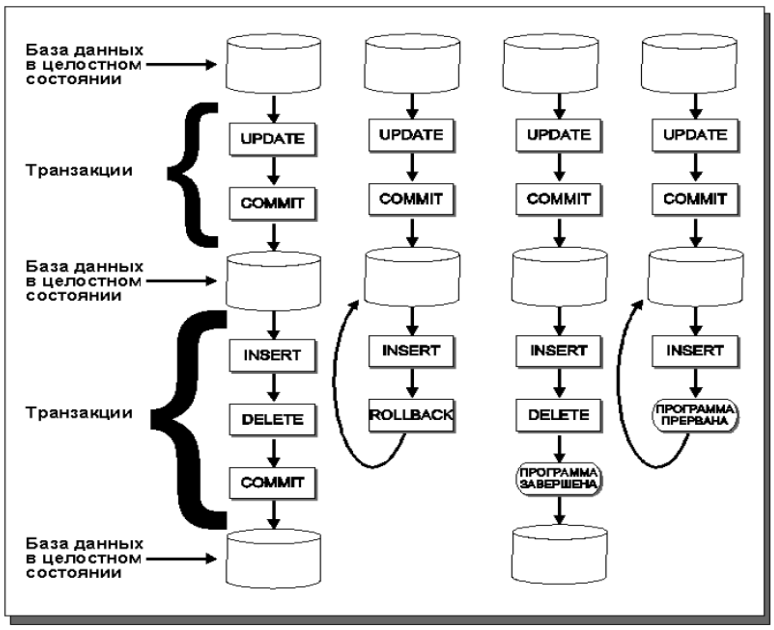



**Рис. 7.3.** Выполненные и отмененные транзакции

Рис. 3.1.1. Подтверждение и отмена транзакций.

Для обработки распределенных транзакций в современных СУБД предусмотрен так называемый **протокол двухфазной фиксации транзакций (two-phase commit)**. Т.е. фиксация распределенной транзакции выполняется в две фазы.

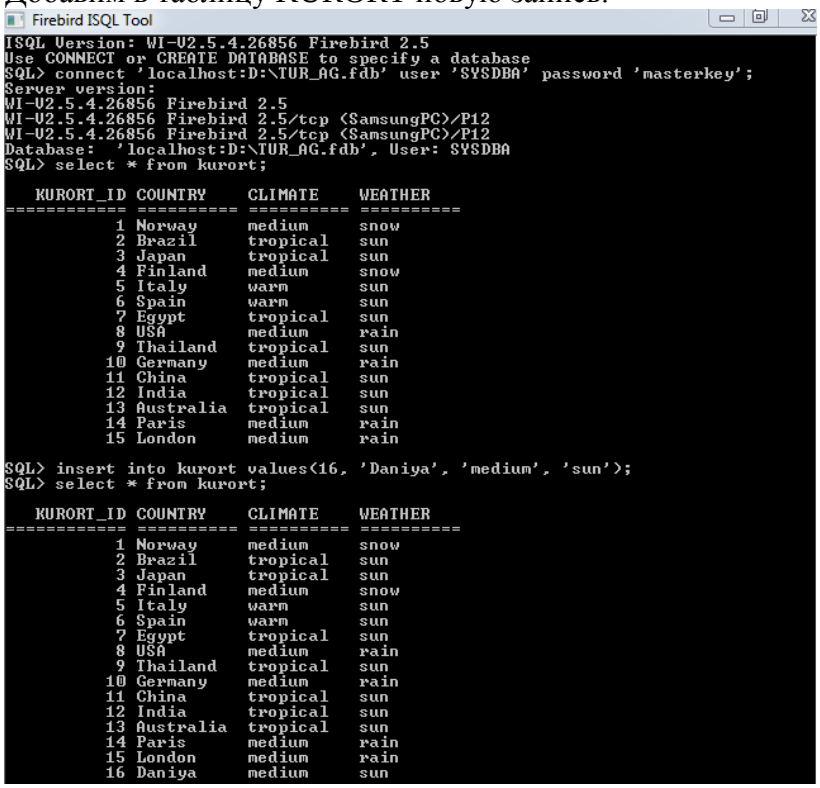
**Фаза 1** начинается, когда при обработке транзакции встретился оператор COMMIT. Сервера переделенной БД (или компонент СУБД, отвечающий за обработку распределенных транзакций) направляет уведомление "подготовиться к фиксации" всем серверам локальных БД, выполняющим распределенную транзакцию. Если все серверы приготовились к фиксации (то есть откликнулись на уведомление, и их отклик был получен), сервер распределенной БД принимает решение о фиксации. Серверы локальных БД остаются в состоянии готовности и ожидают от него команды "зафиксировать". Если хотя бы один из серверов не откликнулся на уведомление в силу каких- либо причин, будь то аппаратная или программная ошибка, то сервер распределенной БД откатывает локальные транзакции на всех узлах, включая даже те, которые подготовились к фиксации и оповестили его об этом.

**Фаза 2 –** сервер распределенной БД направляет команду "зафиксировать" всем узлам, затронутым транзакцией, и гарантирует, что транзакции на них будут зафиксированы. Если связь с локальной базой данных потеряна в интервал времени между моментом, когда сервер распределенной БД принимает решение о фиксации транзакции, и моментом, когда сервер локальной БД подчиняется его команде, то сервер распределенной БД продолжает попытки завершить транзакцию, пока связь не будет восстановлена.

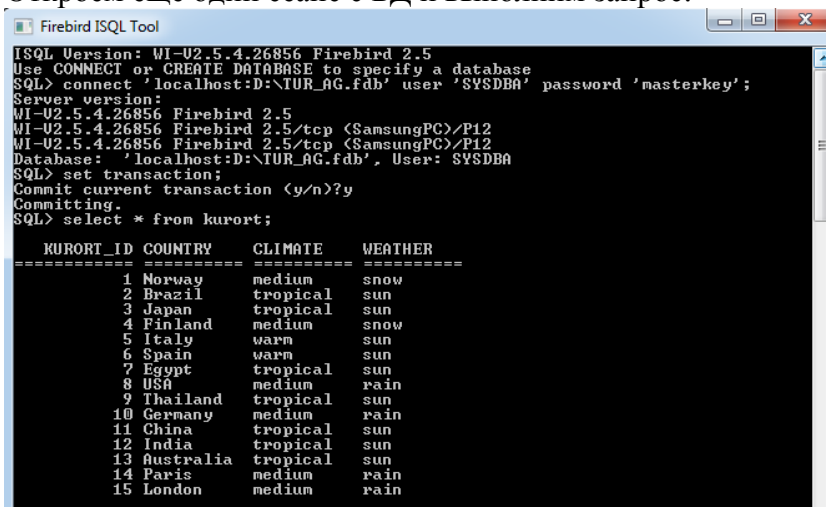
## Выполнение работы

1. Провести эксперименты по запуску, подтверждению и откату транзакций.

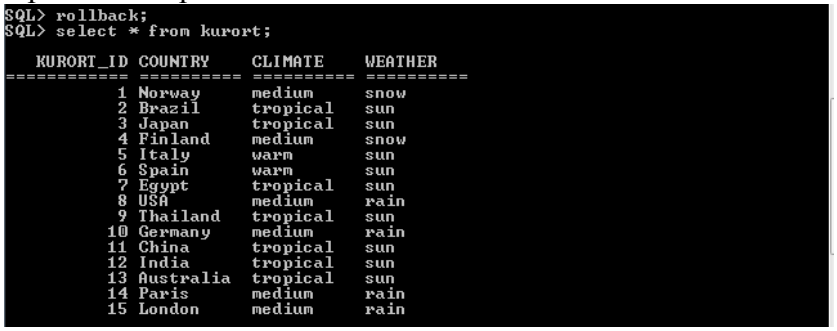
Добавим в таблицу KURORT новую запись.




Откроем еще один сеанс с БД и выполним запрос:




При откате транзакции в обоих сеансах действия по добавлению записей будут отменены.



Снова добавим те же данные, но с подтверждением в первом и посмотрим на результат во втором сеансе:

```
commit;
```

Во втором:

```
set transaction;
```

```
SQL> select * from kurort;

   KURORT_ID COUNTRY    CLIMATE    WEATHER
============ ========== ========== ==========
           1 Norway     medium     snow
           2 Brazil     tropical   sun
           3 Japan      tropical   sun
           4 Finland    medium     snow
           5 Italy      warm       sun
           6 Spain      warm       sun
           7 Egypt      tropical   sun
           8 USA        medium     rain
           9 Thailand   tropical   sun
          10 Germany    medium     rain
          11 China      tropical   sun
          12 India      tropical   sun
          13 Australia  tropical   sun
          14 Paris      medium     rain
          15 London     medium     rain
          16 Daniya     medium     sun
```

2. Разобраться с уровнями изоляции транзакций в Firebird.

   Уровень изолированности транзакции определяет, какие изменения, сделанные в других транзакциях, будут видны в данной транзакции. Каждая транзакция имеет свой уровень изоляции, который устанавливается при ее запуске и остается неизменным в течение всей ее жизни.
   Транзакции в Firebird могут иметь 3 основных возможных уровня изоляции: READ COMMITTED, SNAPSHOT и SNAPSHOT TABLE STABILITY. Каждый из этих трех уровней изоляции определяет правила видимости тех действий, которые выполняются другими транзакциями.
   • READ COMMITTED ( "читать подтвержденные данные"). Уровень изоляции READCOMMITTED используется, когда мы хотим видеть все подтвержденные результаты параллельно выполняющихся (т. е. в рамках других транзакций) действий. Этот уровень изоляции гарантирует, что мы не сможем прочитать неподтвержденные данные, измененные в других транзакциях, и делает
   возможным прочитать подтвержденные данные.
   • SNAPSHOT. Этот уровень изоляции используется для создания "моментального" снимка базы данных. Все операции чтения данных, выполняемые в рамках транзакции с уровнем изоляции SNAPSHOT, будут видеть только состояние базы данных на момент начала запуска транзакции. Все изменения, сделанные в параллельных транзакциях, не видны в этой транзакции. В то же время SNAPSHOT не блокирует данные, которые он не изменяет.
   • SNAPSHOT TABLE STABILITY. Это уровень изоляции также создает "моментальный" снимок базы данных, но одновременно блокирует на запись данные, задействованные в операциях, выполняемые данной транзакцией. Это означает, что если транзакция SNAPSHOT TABLE STABILITY изменила данные в какой-нибудь таблице, то после этого данные в этой таблице уже не могут быть изменены в других параллельных транзакциях. Кроме того, транзакции с уровнем изоляции SNAPSHOT TABLE STABILITY не могут получить доступ к таблице, если данные в них уже изменяются в контексте других транзакций.

**Уровень изоляции READ COMMITTED**
Эксперимент:
1ый терминал:set transaction isolation level read committed;
2ой терминал:insert into kurort(kurort_id, country, climate, weather) values(17, 'Russia', 'medium' 'rain');
1ый терминал:select * from kurort;
2ой терминал:Commit;

**1 сессия:**
SQL> set transaction isolation level read committed;
Commit current transaction (y/n)?y
Committing.
SQL> select * from kurort;

| KURORT_ID | COUNTRY | CLIMATE | WEATHER |
|---|---|---|---|
| 1 | Norway | medium | snow |
| 2 | Brazil | tropical | sun |
| 3 | Japan | tropical | sun |
| 4 | Finland | medium | snow |
| 5 | Italy | warm | sun |
| 6 | Spain | warm | sun |
| 7 | Egypt | tropical | sun |
| 8 | USA | medium | rain |
| 9 | Thailand | tropical | sun |
| 10 | Germany | medium | rain |
| 11 | China | tropical | sun |
| 12 | India | tropical | sun |
| 13 | Australia | tropical | sun |
| 14 | Paris | medium | rain |
| 15 | London | medium | rain |
| 16 | Daniya | medium | sun |
| 17 | Russia | medium | rain |

**2 сессия:**
SQL> insert into kurort(kurort_id, country, climate, weather) values(17, 'Russia
', 'medium', 'rain');
SQL> commit;

Уровень изоляции READ COMMITTED имеет два режима - NO RECORD VERSION и RECORD VERSION. По умолчанию включен первый режим. Проведем тот же эксперимент включив второй режим:

**1 сессия:**
SQL> set transaction isolation level read committed RECORD_VERSION;
Commit current transaction (y/n)?y
Committing.
SQL> select * from kurort;

| KURORT_ID | COUNTRY | CLIMATE | WEATHER |
|---|---|---|---|
| 1 | Norway | medium | snow |

2 Brazil tropical sun
3 Japan tropical sun
4 Finland medium snow
5 Italy warm sun
6 Spain warm sun
7 Egypt tropical sun
8 USA medium rain
9 Thailand tropical sun
10 Germany medium rain
11 China tropical sun
12 India tropical sun
13 Australia tropical sun
14 Paris medium rain
15 London medium rain
16 Daniya medium sun

**2 сессия:**
SQL> insert into kurort(kurort_id, country, climate, weather) values(17, 'Russia
', 'medium', 'rain');

**Уровень изоляции SNAPSHOT**
Эксперимент:
1ый терминал: set transaction isolation level snapshot;
2ой терминал: insert into kurort(kurort_id, country, climate, weather) values(17, 'Russia', 'medium' 'rain');
2ой терминал: commit;
1ый терминал: select * from kurort;

**1 сессия:**
SQL> set transaction isolation level snapshot;
Commit current transaction (y/n)?y
Committing.
SQL> select * from kurort;

KURORT_ID COUNTRY CLIMATE WEATHER
========== ========== ========== ==========
1 Norway medium snow
2 Brazil tropical sun
3 Japan tropical sun
4 Finland medium snow
5 Italy warm sun
6 Spain warm sun
7 Egypt tropical sun
8 USA medium rain
9 Thailand tropical sun
10 Germany medium rain
11 China tropical sun
12 India tropical sun
13 Australia tropical sun
14 Paris medium rain
15 London medium rain
16 Daniya medium sun

**2 сессия:**
SQL> insert into kurort(kurort_id, country, climate, weather) values(17, 'Russia
', 'medium', 'rain');
SQL> commit;

**Уровень изоляции SNAPSHOT TABLE STABILITY**
Эксперимент:
1ый терминал: set transaction isolation level snapshot TABLE STABILITY;
2ой терминал: set transaction isolation level snapshot TABLE STABILITY;
1ый терминал: update kurort set country='RF' where kurort_id=17;
2ой терминал: select * from kurort;
1ый терминал: Commit;
1ый терминал: select * from kurort;
2ой терминал: select * from kurort;

**1 сессия:**
SQL> set transaction isolation level snapshot TABLE STABILITY;
Commit current transaction (y/n)?y
Committing.
SQL> select * from kurort;

| KURORT_ID | COUNTRY | CLIMATE | WEATHER |
|---|---|---|---|
| 1 | Norway | medium | snow |
| 2 | Brazil | tropical | sun |
| 3 | Japan | tropical | sun |
| 4 | Finland | medium | snow |
| 5 | Italy | warm | sun |
| 6 | Spain | warm | sun |
| 7 | Egypt | tropical | sun |
| 8 | USA | medium | rain |
| 9 | Thailand | tropical | sun |
| 10 | Germany | medium | rain |
| 11 | China | tropical | sun |
| 12 | India | tropical | sun |
| 13 | Australia | tropical | sun |
| 14 | Paris | medium | rain |
| 15 | London | medium | rain |
| 16 | Daniya | medium | sun |
| 17 | Russia | medium | rain |

SQL> update kurort set country='RF' where kurort_id=17;
SQL> commit;
SQL> select * from kurort;

| KURORT_ID | COUNTRY | CLIMATE | WEATHER |
|---|---|---|---|
| 1 | Norway | medium | snow |
| 2 | Brazil | tropical | sun |
| 3 | Japan | tropical | sun |
| 4 | Finland | medium | snow |
| 5 | Italy | warm | sun |

6 Spain warm sun
7 Egypt tropical sun
8 USA medium rain
9 Thailand tropical sun
10 Germany medium rain
11 China tropical sun
12 India tropical sun
13 Australia tropical sun
14 Paris medium rain
15 London medium rain
16 Daniya medium sun
17 RF medium rain

**2 сессия:**

SQL> set transaction isolation level snapshot TABLE STABILITY;
Commit current transaction (y/n)?y
Committing.
SQL> select * from kurort;

KURORT_ID COUNTRY CLIMATE WEATHER
========== ========== ========== ==========
1 Norway medium snow
2 Brazil tropical sun
3 Japan tropical sun
4 Finland medium snow
5 Italy warm sun
6 Spain warm sun
7 Egypt tropical sun
8 USA medium rain
9 Thailand tropical sun
10 Germany medium rain
11 China tropical sun
12 India tropical sun
13 Australia tropical sun
14 Paris medium rain
15 London medium rain
16 Daniya medium sun
17 Russia medium rain

SQL> select * from kurort;

KURORT_ID COUNTRY CLIMATE WEATHER
========== ========== ========== ==========
1 Norway medium snow
2 Brazil tropical sun
3 Japan tropical sun
4 Finland medium snow
5 Italy warm sun
6 Spain warm sun
7 Egypt tropical sun
8 USA medium rain
9 Thailand tropical sun

10 Germany medium rain
11 China tropical sun
12 India tropical sun
13 Australia tropical sun
14 Paris medium rain
15 London medium rain
16 Daniya medium sun
17 Russia medium rain

Основное отличие SNAPSHOT TABLE STABILITY от SNAPSHOT в том, что таблица была заблокирована и на чтение.

**Вывод**

Транзакция — группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Одним из наиболее распространённых наборов требований к транзакциям и транзакционным системам является набор ACID (Atomicity, Consistency, Isolation, Durability).

Для поддержания поставленных требований и эффективного баланса производительности в СУБД введены различные системы изолированности транзакций. Ниже рассмотрено соответствие между уровнем изоляции и режимом транзакции:

| Уровни | Режим |
|---|---|
| 0 — Чтение неподтверждённых данных (грязное чтение) | Нет |
| 1 — Чтение подтверждённых данных | READ COMMITTED |
| 2 — Повторяемое чтение | SNAPSHOT |
| 3 — Сериализуемый | SNAPSHOT TABLE STABILITY |