Санкт-Петербургский Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра Компьютерных Систем и Программных Технологий

**ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №3**

Дисциплина: **Базы данных**

# Тема: Транзакции

А.М. Кириллов

А.В. Мяснов

Выполнил студент гр. № 43501/1

Преподаватель

Санкт-Петербург

2016

# Цель 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.

# Ход работы:

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**:

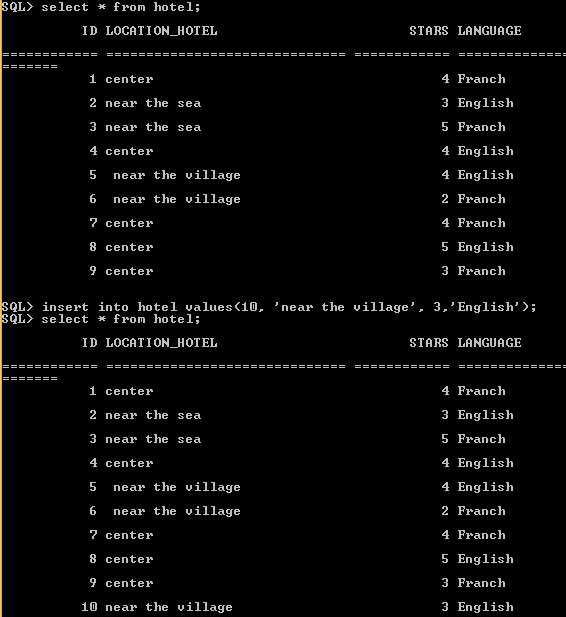
* **Атомарность.** Транзакция выполняется как атомарная операция-либо выполняетсявся транзакция целиком, либо она целиком не выполняется.
* **Согласованность.** Транзакция переводит базу данных из одного согласованного(целостного) состояния в другое согласованное (целостное) состояние. Внутри транзакции 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 должнысохраниться 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**:

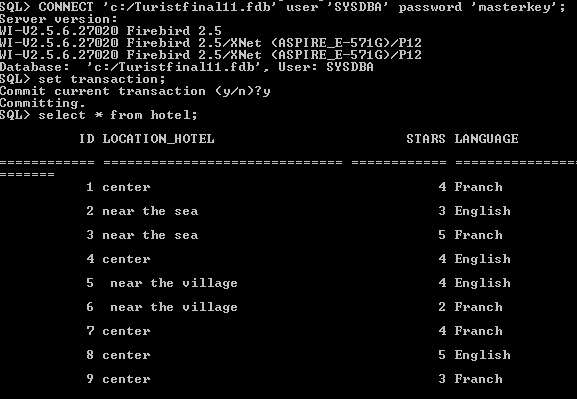
* Подана команда COMMIT WORK (зафиксировать транзакцию).
* Подана команда ROLLBACK WORK (откатить транзакцию).
* Произошло отсоединение пользователя от СУБД.
* Произошел сбой системы.

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

Добавим в таблицу Hotel новую запись:



В новом сеансе БД выполним запрос:

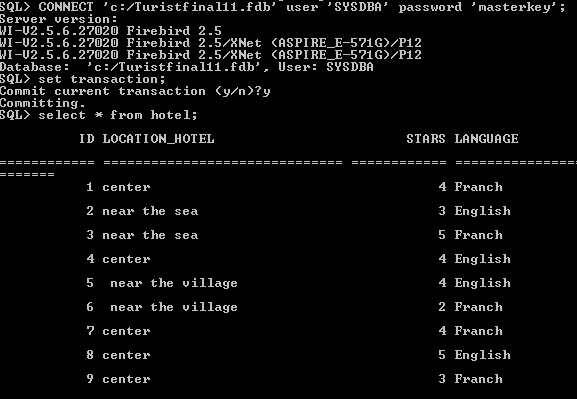


При откате транзакции в обоих сеансах добавление записи в таблицу Hotel была отменена



Cнова добавим новую запись, но уже с подтверждением в первой сессии

Во второй сессии будет:



3. 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 HOTEL (id, location\_hotel, stars, language) VALUES (12, ‘center’, 5,'English');

1ый терминал:select \* from hotel;

2ой терминал:Commit;

1 сессия:

SQL> set transaction isolation level read committed;

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ =======================

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

2 сессия:

SQL> INSERT INTO HOTEL (id, location\_hotel, stars, language) VALUES (12, 'center', 5,

CON> 'English');

SQL> commit;

Проведем тот же эксперимент включив второй режим - RECORD VERSION:

1 сессия

SQL> set transaction isolation level read committed RECORD\_VERSION;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ ========

=======

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

2 сессия:

SQL> INSERT INTO HOTEL (id, location\_hotel, stars, language) VALUES (13, 'near tje sea', 5,

CON> 'English');

**Уровень изоляции SNAPSHOT**

Эксперимент:

1ый терминал: set transaction isolation level snapshot;

2ой терминал: INSERT INTO HOTEL (id, location\_hotel, stars, language) VALUES (14, 'near the ocean', 2,'English');

2ой терминал: commit;

1ый терминал: select \* from hotel;

1 Сессия:

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ ========

=======

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

13 near tje sea 5 English

2 Сессия:

SQL> INSERT INTO HOTEL (id, location\_hotel, stars, language) VALUES (14, 'near the ocean', 2,'English');

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 hotel set location\_hotel='Tuta' where id=13;

2ой терминал: select \* from hotel;

1ый терминал: Commit;

1ый терминал: select \* from hotel;

2ой терминал: select \* from hotel;

1 Сессия:

SQL> set transaction isolation level snapshot TABLE STABILITY;

Commit current transaction (y/n)?y

Committing.

SQL> update hotel set location\_hotel='Tuta' where id=13;

SQL> commit;

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ ========

=======

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

13 Tuta 5 English

14 near the ocean 2 English

2 Сессия:

SQL> set transaction isolation level snapshot TABLE STABILITY;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ ========

=======

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

13 near tje sea 5 English

14 near the ocean 2 English

SQL> select \* from hotel;

ID LOCATION\_HOTEL STARS LANGUAGE

============ ============================== ============ ========

=======

1 center 4 Franch

2 near the sea 3 English

3 near the sea 5 Franch

4 center 4 English

5 near the village 4 English

6 near the village 2 Franch

7 center 4 Franch

8 center 5 English

9 center 3 Franch

10 near the village 3 English

11 near the village 4 English

12 center 5 English

13 near tje sea 5 English

14 near the ocean 2 English

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).