**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по лабораторной работе №7**

**«Изучение механизма транзакций»**

**Информационное обеспечение систем управления**

Студент гр. 43501/1 Красильников В.В.

Преподаватель Мяснов А.В.

Санкт-Петербург

2015

# 1. Цель работы

Ознакомиться 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.

# 3. Запуск, подтверждение и откат транзакций

Добавим запись в таблицу:

Use CONNECT or CREATE DATABASE to specify a database

SQL> connect 'localhost:C:/dbnew/FLAT6.fdb' user 'SYSDBA' password 'masterkey';

Database: 'localhost:C:/dbnew/FLAT6.fdb', User: SYSDBA

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

Получаем список районов, который в данный момент насчитывает 7 позиций.

Добавим 8ой район.

SQL> insert into DISTRICTS values (8,'Petrogradskij');

Создадим еще один сеанс связи с БД и просмотрим ту же таблицу:

Use CONNECT or CREATE DATABASE to specify a database

SQL> connect 'localhost:C:/dbnew/FLAT6.fdb' user 'SYSDBA' password 'masterkey';

Database: 'localhost:C:/dbnew/FLAT6.fdb', User: SYSDBA

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

Новая запись в этом запросе отсутствует, так как транзакция, в которой она была добавлена, не подтверждена.

Выполним подтверждение во 2ом сеансе и посмотрим таблицу:

SQL> commit;

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

Новая запись появилась, так как данные получены после подтверждения транзакции.

# 4. Уровни изоляции транзакций

Уровень изоляции 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 три 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.

## 4.1 Уровень изоляции READ COMMITTED

1ая сессия:

SQL> set transaction isolation level read committed;

Committing.

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

2ая сессия:

SQL> insert into DISTRICTS values (9,'Kolpinskij');

1ая сессия:

SQL> select \* from DISCTRICTS;

Выполнение запроса не происходит.

2ая сессия:

SQL> commit;

Запрос на чтение выполнен:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

Запрос на чтение неподтвержденных данных был заблокирован до подтверждения транзакции.

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

1ая сессия:

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

SQL> insert into model values (9,'2109',3);

2ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

1ая сессия:

SQL> commit;

2ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

Данный режим позволяет сразу же читать данные из последней подтвержденной транзакции.

## 4.2 Уровень изоляции SNAPSHOT

1ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

2ая сессия:

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

1ая сессия:

SQL> insert into DISTRICTS values (11,'Primorskij');

SQL> commit;

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Primorskij

2ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

SQL> commit;

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Primorskij

Для чтения измененных данных потребовалось завершить транзакцию.

Изменим эксперимент, добавив одновременное чтение:

2ая сессия:

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

1ая сессия:

SQL> update DISTRICTS set DISTRICTNAME = 'Moskovskij' where ID\_DISTRICT = 11;

2ая сессия:

SQL> update DISTRICTS set DISTRICTNAME = 'Moskovskij' where ID\_DISTRICT = 11;

Операция не выполняется

1ая сессия:

SQL> commit;

2ая сессия:

Statement failed, SQLSTATE = 40001

deadlock

-update conflicts with concurrent update

-concurrent transaction number is 4125

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Primorskij

SQL> commit;

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Moskovskij

1ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Moskovskij

Попытка изменения данных, которые уже изменяются в другой транзакции, при уровне изоляции SNAPSHOT вызывает deadlock.

## 4.3 Уровень изоляции SNAPSHOT TABLE STABILITY

1ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Moskovskij

2ая сессия:

SQL> set transaction isolation level snapshot table stability;

Commit current transaction (y/n)?y

Committing.

1ая сессия:

SQL> set transaction isolation level snapshot table stability;

Commit current transaction (y/n)?y

Committing.

2ая сессия:

SQL> update DISTRICTS set DISTRICTNAME = 'Kronshtadtskij' where ID\_DISTRICT = 11;

1ая сессия:

SQL> select \* from DISTRICTS;

Ожидание

2ая сессия:

SQL> commit;

В 1ой сессии получаем ранее введены запрос после ожидания

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Moskovskij

2ая сессия:

SQL> select \* from DISTRICTS

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Kronshtadtskij

SQL> commit;

1ая сессия:

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Moskovskij

SQL> commit;

SQL> select \* from DISTRICTS;

ID\_DISTRICT DISTRICTNAME

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

1 Nevskij

2 Kurortnij

5 Viborgskij

3 Petrodvorcovij

4 Krasnoselskij

6 Centralnij

7 Lomonosovskij

8 Petrogradskij

9 Kolpinskij

10 Frunzenskij

11 Kronshtadtskij

Отличие от уровня изоляции SNAPSHOT в том, что попытка чтения измененных данных также вызывает блокировку.

# 5. 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.

Существует 4 стандартных уровня (ANSI SQL-92) изолированности транзакций: Dirty Read – чтение неподтвержденных данных, Read Commited - невоспроизводимое (или неповторяемое) чтение, Repeatable Read - воспроизводимое (или повторяемое) чтение, транзакция видит только те данные, которые существовали на момент ее старта, Serialized - транзакции выполняются так, как будто никаких других транзакций в этот момент не существует.

Благодаря транзакциям обеспечивается целостность состояния БД при переходе от одного состояния к другому.