**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 Матлаш И.Г.

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 'C:/db/as.fdb' user 'SYSDBA' password 'masterkey';

Database: 'C:/db/as.fdb', User: SYSDBA

SQL> insert into model values (7,'2111',3);

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

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

ISQL Version: WI-V2.5.3.26780 Firebird 2.5

Use CONNECT or CREATE DATABASE to specify a database

SQL> connect 'C:/db/as.fdb' user 'SYSDBA' password 'masterkey';

Server version:

WI-V2.5.3.26780 Firebird 2.5

WI-V2.5.3.26780 Firebird 2.5/XNet (DEMON)/P12

WI-V2.5.3.26780 Firebird 2.5/XNet (DEMON)/P12

Database: 'C:/db/as.fdb', User: SYSDBA

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

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

Выполним подтверждение в 1ом сеансе:

SQL> commit;

Проверим таблицу во втором сеансе:

SQL> rollback;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

Новая запись появилась, так как данные получены после подтверждения транзакции в 1 сеансе.

# 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

Выделение текста: 1 сессия, 2 сессия.

SQL> set transaction isolation level read committed;

Commit current transaction (y/n)?y

Committing.

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

SQL> insert into model values (8,'2112',3);

SQL> select \* from model;

Выполнение запроса не происходит

SQL> commit;

Запрос на чтение выполнен:

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

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:

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

Commit current transaction (y/n)?y

Committing.

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

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

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

## 4.2 Уровень изоляции SNAPSHOT

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

SQL> set transaction isolation level snapshot;

Commit current transaction (y/n)?y

Committing.

SQL> insert into model values (10,'2105',3);

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2105 3

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

Получены неизмененные данные

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 Granta 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2105 3

Для чтения измененных данных потребовалось завершить транзакцию.

Изменим эксперимент, добавив одновременное чтение:

SQL> set transaction isolation level snapshot;

SQL> update model set name='2100' where id=10;

SQL> update model set name='2101' where id=10;

Операция не выполняется

SQL> commit;

Statement failed, SQLSTATE = 40001

deadlock

-update conflicts with concurrent update

-concurrent transaction number is 3405

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2105 3

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2100 3

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2100 3

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

## 4.3 Уровень изоляции SNAPSHOT TABLE STABILITY

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2100 3

SQL> set transaction isolation level snapshot table stability;

Commit current transaction (y/n)?y

Committing.

SQL> set transaction isolation level snapshot table stability;

Commit current transaction (y/n)?y

Committing.

SQL> update model set name='Kalina' where id = 10;

SQL> select \* from model;

Ожидание

SQL> commit;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2100 3

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 Kalina 3

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 2100 3

SQL> commit;

SQL> select \* from model;

ID NAME MARK

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

1 911 Carrera 2

2 GL450 1

3 2110 3

4 Priora 3

5 2106 3

6 2104 3

7 2111 3

8 2112 3

9 2109 3

10 Kalina 3

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.