Санкт-Петербургский Государственный Политехнический Университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

**ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №7**

Дисциплина: **Базы данных**

Тема: **Изучение механизма транзакций**

Н.В. Тышковец

А.В. Мяснов

Выполнил студент гр. № 43501/1

Преподаватель

Санкт-Петербург

2015

## Цели 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.

**Ход работы**

*Уровни изоляции транзакций*

Уровень изоляции транзакции определяет, могут ли другие (конкурирующие) транзакции вносить изменения в данные, измененные текущей транзакцией, а также может ли текущая транзакция видеть изменения, произведенные конкурирующими транзакциями и наоборот.

Транзакции в Firebird могут иметь 3 основных возможных уровня изоляции:

*READ COMMITTED* используется, когда мы желаем видеть все подтвержденные результаты параллельно выполняющихся (т. е. в рамках других транзакций) действий. Используя этот уровень, мы не сможем прочитать неподтвержденные данные, измененные в других транзакциях.

*SNAPSHOT* используется для создания "моментального" снимка базы данных. Все изменения, сделанные в параллельных подтвержденных (и неподтвержденных) транзакциях, не видны в этой транзакции. В то же время SNAPSHOT не блокирует данные, которые он не изменяет.

*SNAPSHOT TABLE STABILITY* создает "моментальный" снимок базы данных, но одновременно блокирует на запись данные, задействованные в операциях, выполняемые данной транзакцией. 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.

Уровень изоляции задаётся командой

set transaction isolation level <название\_уровня>;

Проверим уровень изоляции READ COMMITTED. Для этого в первом сеансе связи с БД (обычном) выполним команду:

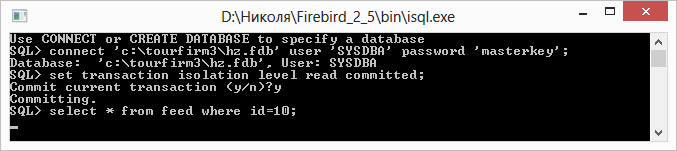
insert into feed(id, feed\_type) values (10, ‘hello feed’);

Не будем делать commit, а откроем вместо этого второй сеанс связи с базой, с уровнем изоляции READ COMMITTED, выполнив команду:

set transaction isolation level READ COMMITTED;

Сделаем выборку из таблицы feed\_type значения с id=10, выполнив команду

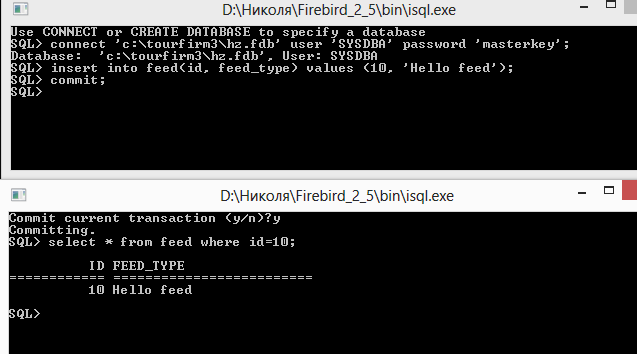
select \* from feed where id=10;



*рис.1. Окно консольного режима связи с БД №2*

Транзакция перешла в режим ожидания, результат не отображается.

Как только в первом сеансе связи выполним команду commit, результат выборки появится во втором сеансе связи:



*рис.2. Сеансы связи с БД*

Рассмотрим уровень изоляции SNAPSHOT.

Для этого во втором соединении открываем транзакцию с уровнем изоляции SNAPSHOT, выполняя команду

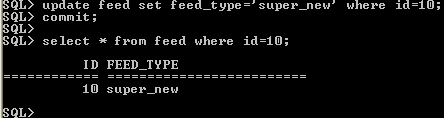
set transaction isolation level SNAPSHOT;

В первом соединении обновим только что созданную строчку с id=10, выполнив команду

update feed SET feed\_type='super\_new' where id=10;

commit;

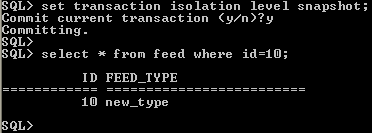
Обратим внимание, что commit был выполнен. Посмотрим результат, выполнив select:



*рис.3. Окно соединения №2*

Видим, что результат обновился.

Однако в соединении №2, при выполнении этой же команды select результат остался прежним, несмотря на то, что был выполнен commit:



*рис.4. Окно соединения №2*

Т.е. мы экспериментально подтвердили тот факт, что при транзакции SNAPSHOT все изменения, сделанные в параллельных соединениях, не видны.

Рассмотрим изоляционный уровень SNAPSHOT TABLE STABILITY.

Во втором соединении создадим транзакцию с изолирующим уровнем SNAPSHOT TABLE STABILITY. Тут же внесём изменение в ту же самую запись, что и выше, выполнив команду

update feed SET feed\_type='super\_puper\_new' where id=10;

Из первого соединения пытаемся менять ту же строку:

update feed SET feed\_type='super\_puper\_new111' where id=10;

Транзакция вошла в режим ожидания, не завершившись:



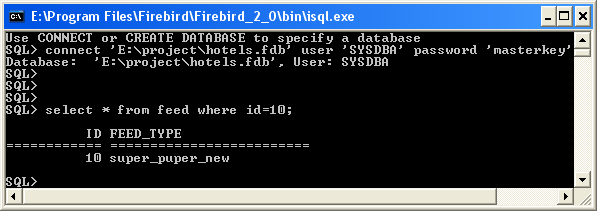
*рис.5. Окно соединения №1*

Теперь завершим транзакцию в соединении №2, набрав команду commit. В соединении №1 вместо ожидания пришла ошибка:



*рис.6. Окно соединения №1*

В итоге убедились, что обновление, вносимое в соединении №1, не вступило в силу. Проверили это, закрыв предыдущие соединения и открыв абсолютно новое:



*рис.7. Новое соединение*

*Вывод:*

## В ходе выполнения данной лабораторной работы мы познакомились с механизмом транзакций и возможностями ручного управления транзакциями. Также рассмотрели в теории и практике различные уровни изоляции транзакций, предусмотренные в Firebird.

Было экспериментально доказано, что используя уровень изоляции *READ COMMITTED* нельзя прочитать неподтвержденные данные, измененные в других транзакциях. Это наиболее часто используемый режим. При уровне SNAPSHOT все изменения, сделанные в параллельных соединениях, не видны. А используя изоляционный уровень SNAPSHOT TABLE STABILITY данные в таблице уже не могут быть изменены в других параллельных транзакциях. Данный режим используется редко и для коротких транзакций.

Механизм транзакции незаменим при работе с крупными базами данных. Он позволяет поддерживать ***логическую целостность данных*** при параллельной работе нескольких клиентов с базой данных, а так же при сбоях.

Транзакция обладает следущими свойствами:

*атомарность* - выполнение или не выполнение всех DML команд входящих в тразакцию; *целостность БД* - завершение транзакции не должно нарушать целостность БД;

*изоляция* - можно отображать либо исходные данные, которые были до начала транзакции, либо новые данные после выполнения транзакции;

*сохранность данных* - если пользователю пришло подтверждение выполнения транзакции, то его изменения не будут отменены по каким-либо причинам.

Из минусов можно отметить то, что сервер с поддержкой транзакций БД требует больше памяти и прочих ресурсов. Также можно отметить более низкую скорость (при прочих равных условиях). Другими словами, есть выбор: работа будет вестись чуть медленнее, но всё будет надёжно работать, без транзитивной же системы – чуть быстрее, но явно ненадёжно. Поэтому преимущества транзакций неоспоримы.