Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №7

По дисциплине «Базы данных»

«Изучение работы транзакций»

Работу выполнили студенты группы №43501/4 Н.С. Шаляпин\_\_\_\_\_\_\_\_\_

Работу принял преподаватель А.В. Мяснов\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2015

1.Цель работы

Познакомить студентов с механизмом транзакций, возможностями ручного управления транзакциями, уровнями изоляции транзакций.

2.Транзакции

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. 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.
* операции выборки данных SELECT. Любая попытка изменения данных в контексте такой транзакции приведёт 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 неподтверждённые 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 путём её 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, внесённые некоторым оператором, будут видны всем последующим операторам, запущенным в рамках этой же транзакции, независимо от её уровня изолированности. Изменения, произведённые в рамках другой транзакции остаются невидимыми для текущей транзакции до тех пор пока они не подтверждены. Уровень изолированности, а иногда, другие атрибуты, определяет, как транзакции будут взаимодействовать с другой транзакцией, которая хочет подтвердить изменения.

3.Программа работы

* Изучить основные принципы работы транзакций.
* Провести эксперименты но запуску, подтверждению и откату транзакций.
* Разобраться с уровнями изоляции транзакций в Firebird.
* Спланировать и провести эксперименты, показывающие основные возможности тран­закций с различным уровнем изоляции.
* Продемонстрировать результаты преподавателю, ответить на контрольные вопросы.

4.Ход работы

1. Были изучены основные принципы работы транзакций.
2. Были проведены эксперименты но запуску, подтверждению и откату транзакций:

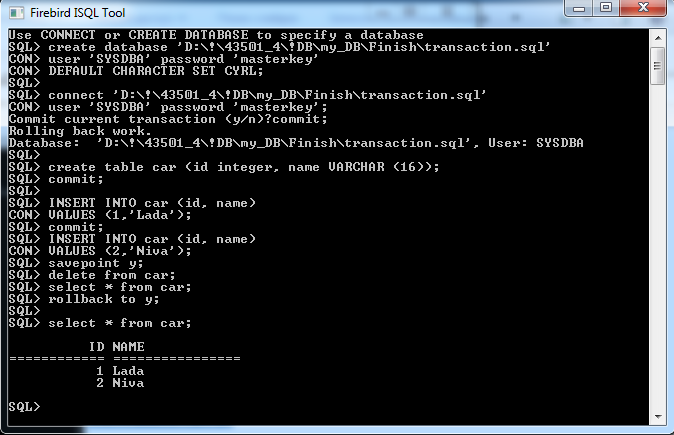


Рис. 1: Опыты с подтверждением и запуском транзакций.

1. Были изучены уровни изоляции транзакций в Firebird.
2. Были проведены эксперименты с различными уровнями изоляции транзакций.

Опыты с уровнем изоляции snapshot: позволяет видеть только те изменения, фикса­ция которых произошла не позднее момента старта этой транзакции. Любые подтвер­ждённые изменения, сделанные другими конкурирующими транзакциями, не будут видны в такой транзакции в процессе ее активности без её перезапуска (рисунок 2).

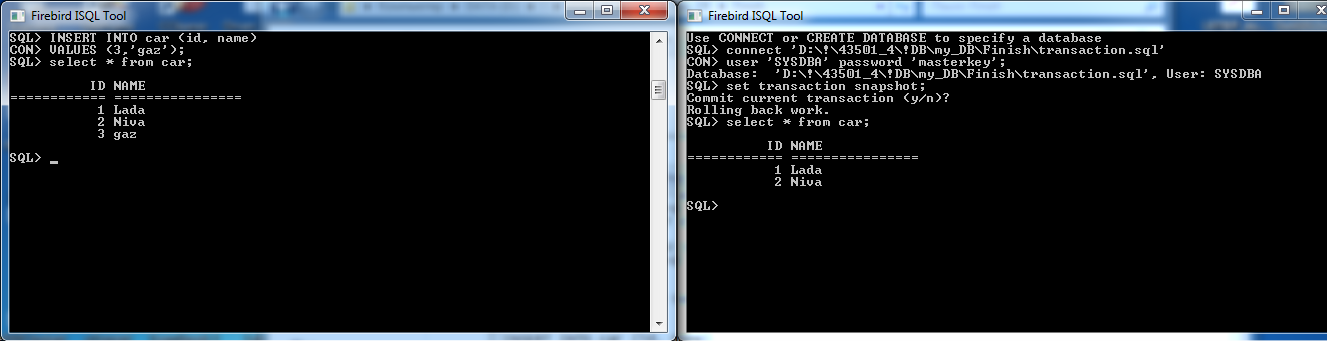


Рис. 2: Два клиента: один выполнил вставку в таблицу, второй при этом не видит произ­веденных изменений.

Опыты с уровнем изоляции snapshot table stability: позволяет видеть только те изменения, фиксация которых произошла не позднее момента старта этой транзакции. При этом после старта такой транзакции в других клиентских транзакциях невозможно выполнение изменений ни в каких таблицах этой базы данных, уже каким- либо образом измененных первой транзакцией (рисунки 3-5).

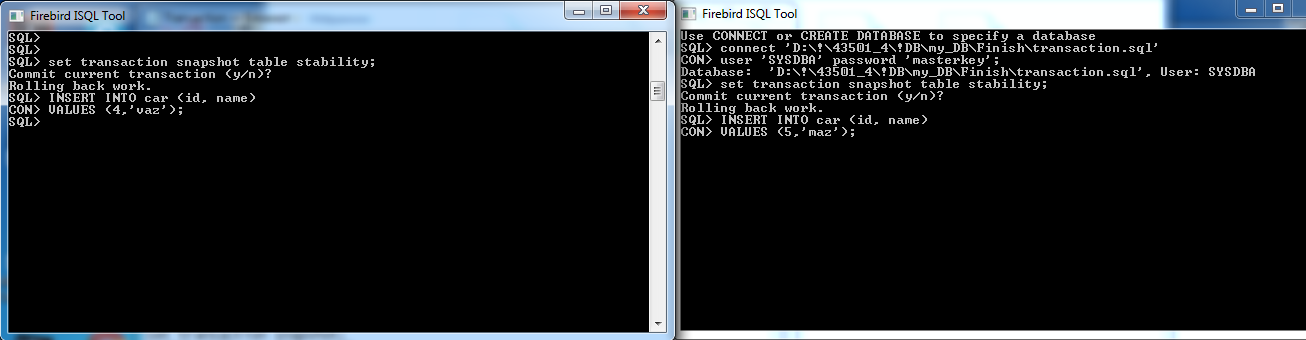


Рис. 3: Два клиента: один выполни вставку в таблицу, второй при этом но может завер­шить операцию вставки.

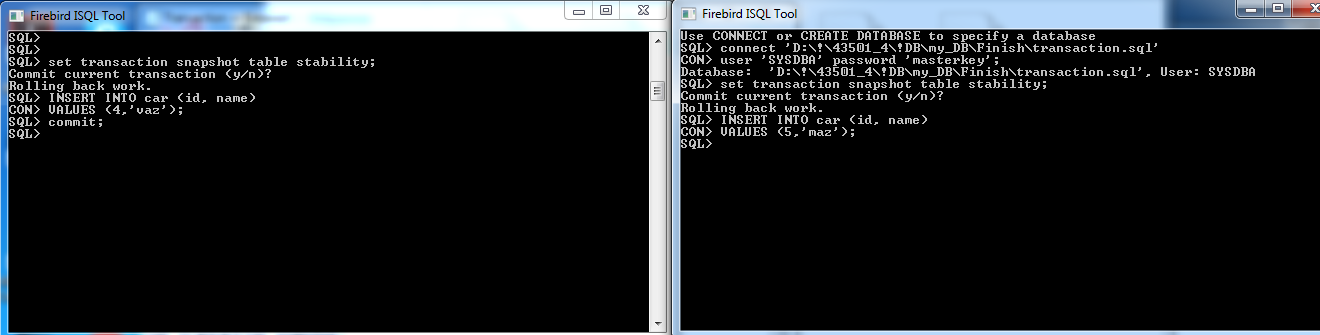


Рис. 4: Два клиента: после того, как первый клиент зафиксировал изменения, второй смог выполнить операцию вставки.

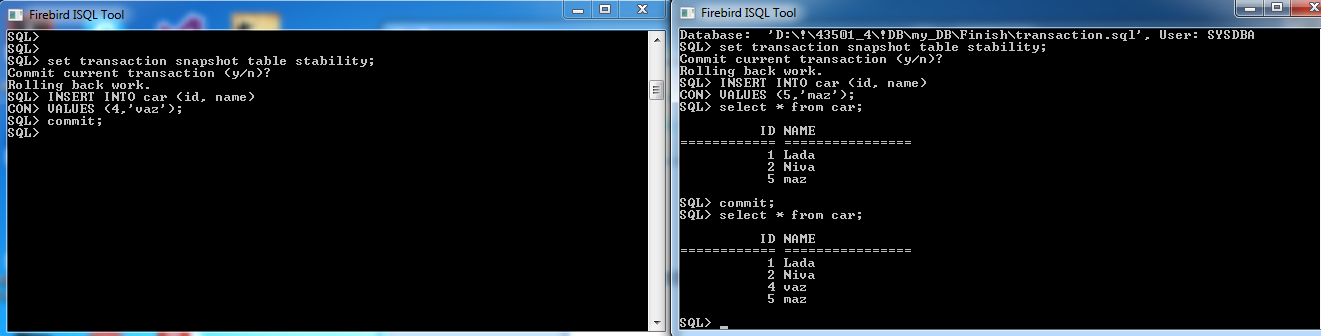
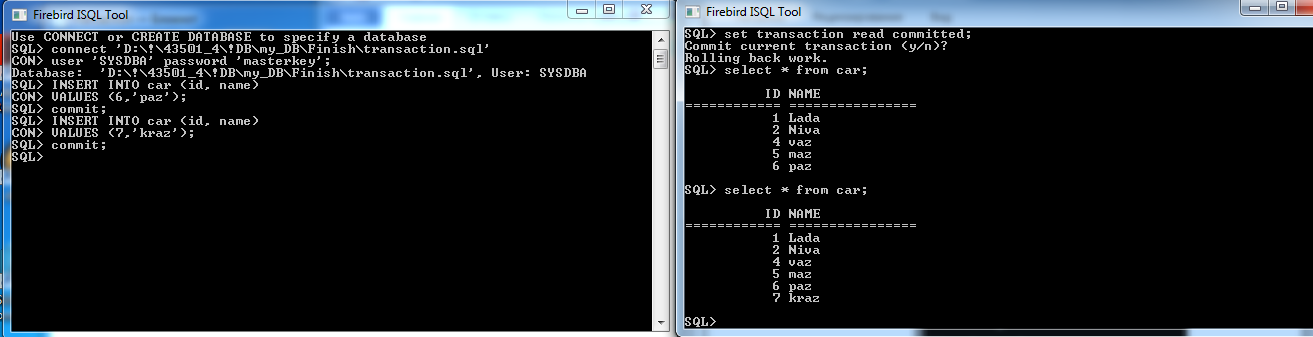


Рис. 5: Два клиента: изменения, произведенные в других транзакциях видны только после того, как изменения были зафиксированы.

Опыты с уровнем изоляции read committed: позволяет в транзакции без её переза­пуска видеть все подтверждённые изменения данных базы данных, выполненные в других параллельных транзакциях. Неподтверждённые изменения не видны в тран­закции и этого уровня изоляции.

С опцией record\_version: транзакция всегда читает последнюю подтверждённую версию записей таблиц, независимо от того, существуют ли изменённые и ещё не подтверждённые версии этих записей. В этом случае режим разрешения блокировок (WAIT или XO WAIT) никак повлияет па поведение транзакции при её старте (рисунок 6).



Рис, 6: Два клиента: один выполнил вставку в таблицу, второй видит ее сразу после коммита.

С опцией **no record\_version wait:** транзакция всегда читает последнюю подтвер­ждённую версию записей таблиц, независимо от того, существуют ли изменённые и ещё не подтверждённые версии этих записей. В этом случае режим разрешения блокировок (WAIT или NO WAIT) никак не влияет на поведение транзакции при её старте (рисунки 7,8).

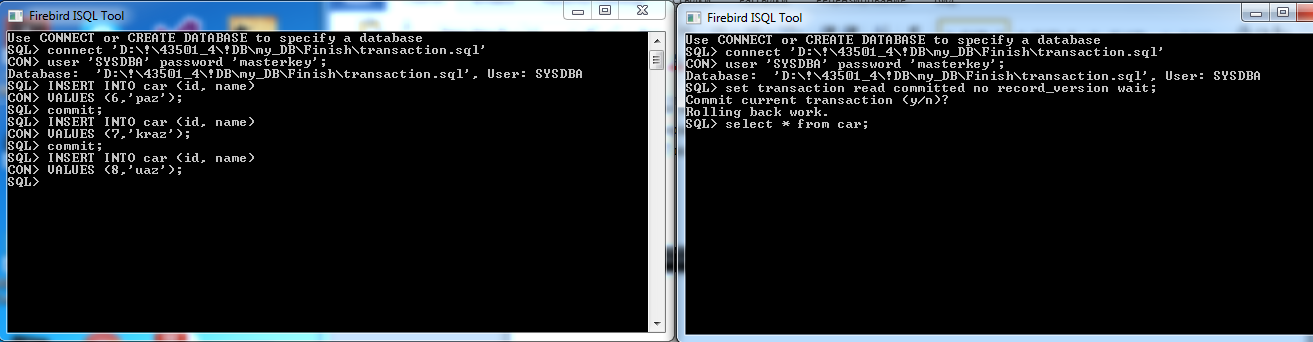


Рис. 7: Два клиента: один выполнил вставку в таблицу, второй не может выполнить select пока первый не закоммитит.

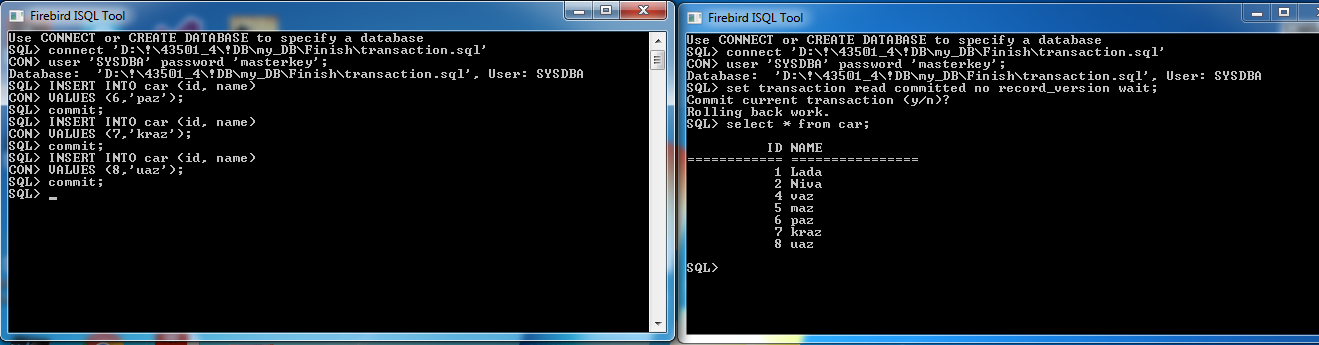


Рис. 8: Два клиента: первый зафиксирован изменения, второй сразу завершил выполнение select.

Сопцией no record\_version no wait: при обращении к таблице, измененной в дру­гой неподтвержденной транзакции, база выбросит исключение (рисунок 9).

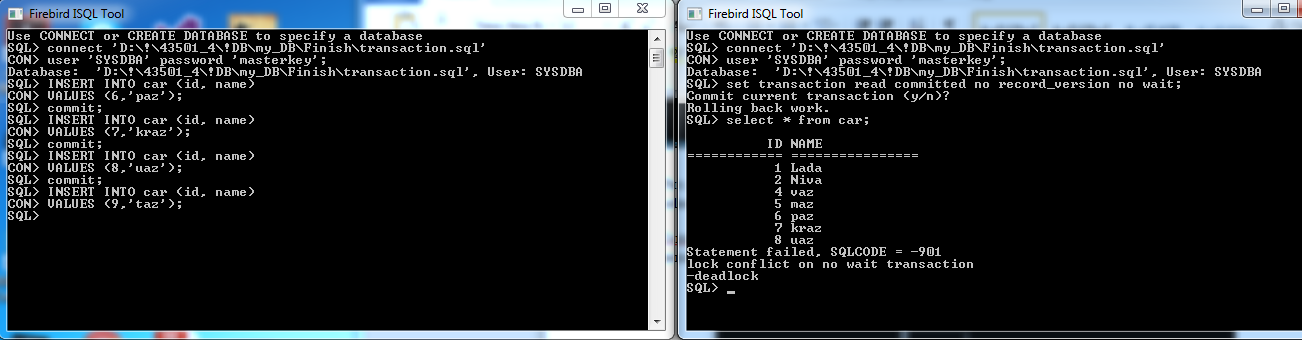


Рис. 9: Два клиента: один выполнил вставку в таблицу, у второго вылетело исключение при обращении к этой таблице.

5.Вывод

В данной лабораторной работе были изучены основные принципы работы транзакций. Всё в Firebird выполняется в рамках транзакций. Транзакция — логическая единица изолированной работы группы последовательных операций над базой данных. Изменения над данными остаются обратимыми до тех пор, пока клиентское приложение не выдаст серверу инструкцию COMMIT.

Транзакция - это неделимая, с точки зрения воздействия на СУБД, последовательность операций манипулирования данными, выполняющаяся по принципу "все или ничего", и переводящая базу данных из одного целостного состояния в другое целостное состояние. Транзакция обладает четырьмя важными свойствами, известными как свойства АСИД:

(А) Атомарность.(С)Согласованность.(И)Изоляция.(Д) Долговечность.

База данных находится в согласованном состоянии, если для этого состояния выполнены все ограничения целостности.

Ограничение целостности - это некоторое утверждение, которое может быть истинным

или ложным в зависимости от состояния базы данных.