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»
«Транзакции»

Работу выполнил студент группы №43501/4 Климова Д.А. _____

Работу принял преподаватель_____ Мяснов А.В._____

Санкт-Петербург
2015

1. Цель работы

Познакомить студентов 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.
5. Продемонстрировать результаты преподавателю, ответить на контрольные вопросы.

3. Ход работы

1. Изучены основные принципы работы транзакций.
2. Эксперименты по запуску, подтверждению и откату транзакций.

```
SQL> select * from books;

    BOOKID BOOKNAME

                        PRICE      RATING      AMOUNT
============ ============================================================================
============ ============ ============ ============
         1 FirstBook

                          230           7          15
         2 SecondBook

                          160           9           4
         3 ThirdBook

                          187           5          23

SQL> commit;
SQL> insert into books values (4,'ForthBook', 198,6,17);
SQL> savepoint y;
SQL> delete from books where bookid = 4;
SQL> select * from books;

    BOOKID BOOKNAME

                        PRICE      RATING      AMOUNT
============ ============================================================================
============ ============ ============ ============
         1 FirstBook

                          230           7          15
         2 SecondBook

                          160           9           4
         3 ThirdBook

                          187           5          23

SQL> rollback to y;
SQL> select * from books;

    BOOKID BOOKNAME
```

```
                        PRICE       RATING      AMOUNT
============ ============================================================
============ ============ ============ ============
        1 FirstBook

                        230         7           15
        2 SecondBook

                        160         9           4
        3 ThirdBook

                        187         5           23
        4 ForthBook

                        198         6           17

SQL> rollback;
SQL> select * from books;

    BOOKID BOOKNAME

                        PRICE       RATING      AMOUNT
============ ============================================================
============ ============ ============ ============
        1 FirstBook

                        230         7           15
        2 SecondBook

                        160         9           4
        3 ThirdBook

                        187         5           23

SQL>
```

В данном эксперименте в уже существующую базу данных была добавлена строка и после этого создана точка сохранения.

После этого созданная строка была удалена.

Вернувшись к точке восстановления мы снова увидели удаленную строку. А после использования rollback; произошел откат к моменту, когда эта строка еще не была создана.

3. Были изучены уровни изоляции по официальной документации firebird.
4. Эксперименты с разными уровнями изоляции:

Уровень изолированности определяет, как транзакции будут взаимодействовать с другой транзакцией, которая хочет подтвердить изменения.

SNAPSHOT:

Уровень изолированности SNAPSHOT (уровень изолированности по умолчанию) означает, что этой транзакции видны лишь 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. Чтобы увидеть эти изменения, нужно завершить транзакцию (подтвердить её или выполнить полный откат, но не откат на точку сохранения) и запустить транзакцию заново.

Запустим двух клиентов, работающих с БД одновременно.

В таблицу books будем добавлять разные данные из разных клиентов.

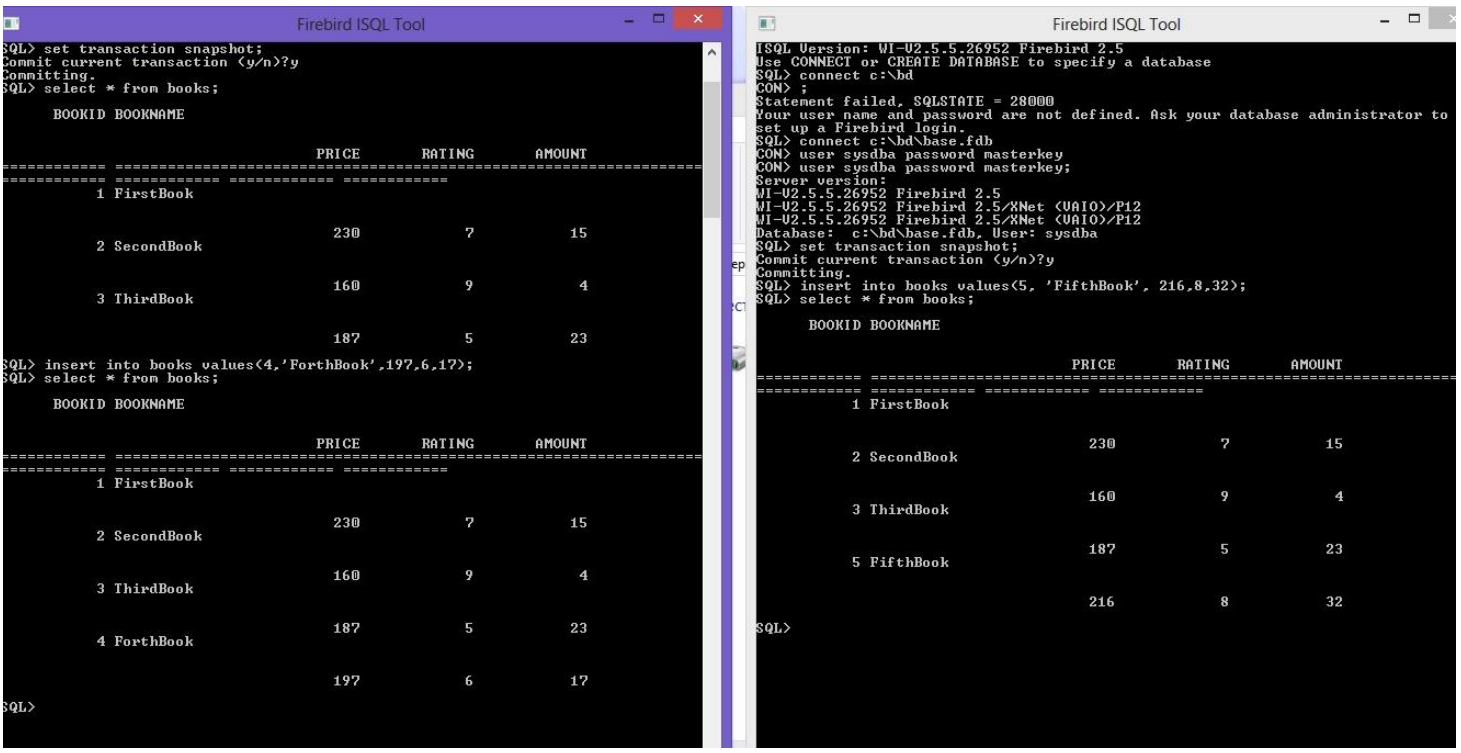


Рис.1. SNAPSHOT: Два параллельных клиента добавляют разные данные в одну таблицу.

Видно что оба клиента работают независимо от другого. Каждый видит только свои изменения.

Завершим транзакции.

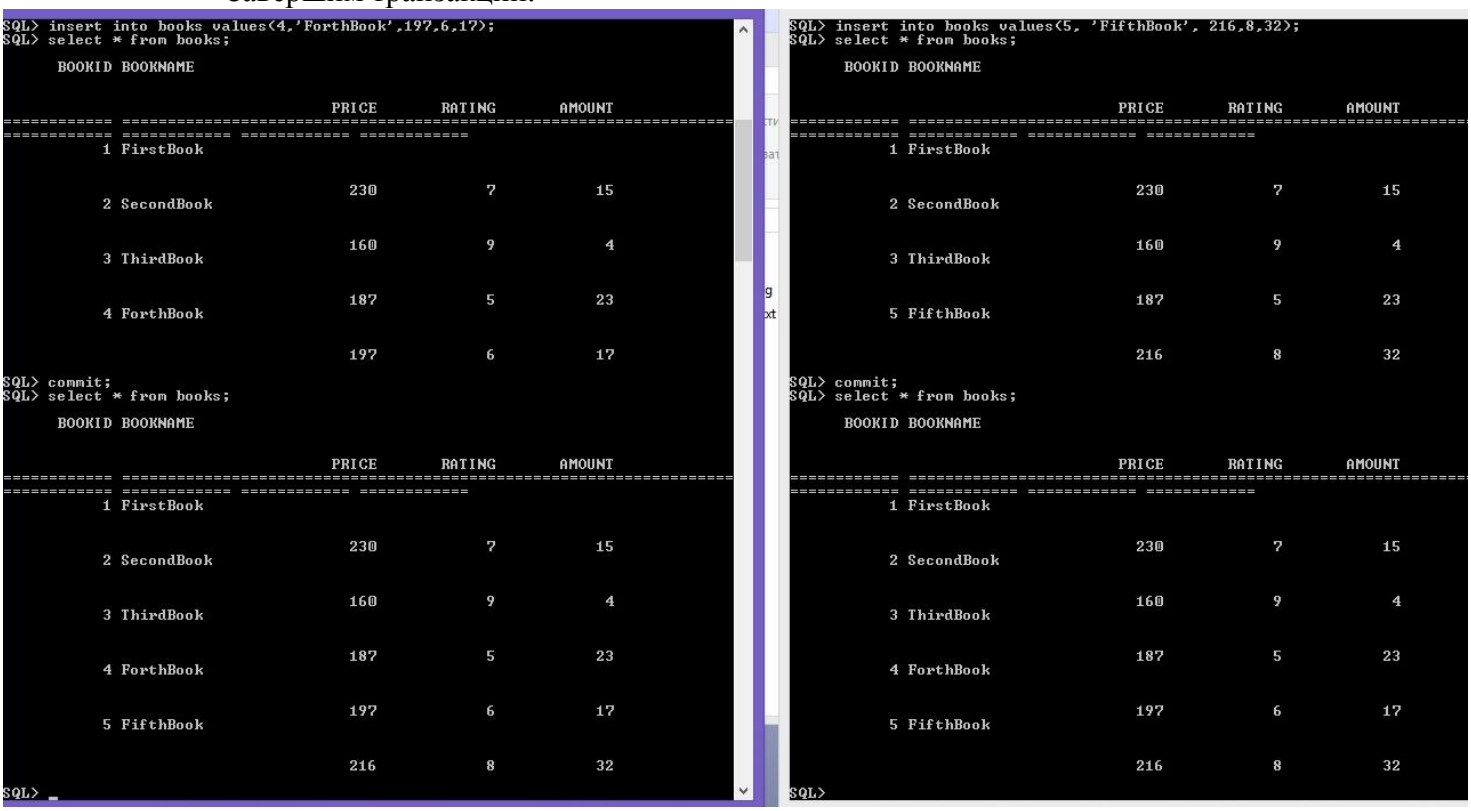


Рис.2. SNAPSHOT: Оба клиента завершают транзакции

SNAPSHOT TABLE STABILITY:

Уровень изоляции транзакции SNAPSHOT TABLE STABILITY позволяет, как и в случае SNAPSHOT, также 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. Все такие попытки в параллельных транзакциях приведут к исключениям базы данных.

Используя два клиента, попробуем добавлять из них данные в одну таблицу.

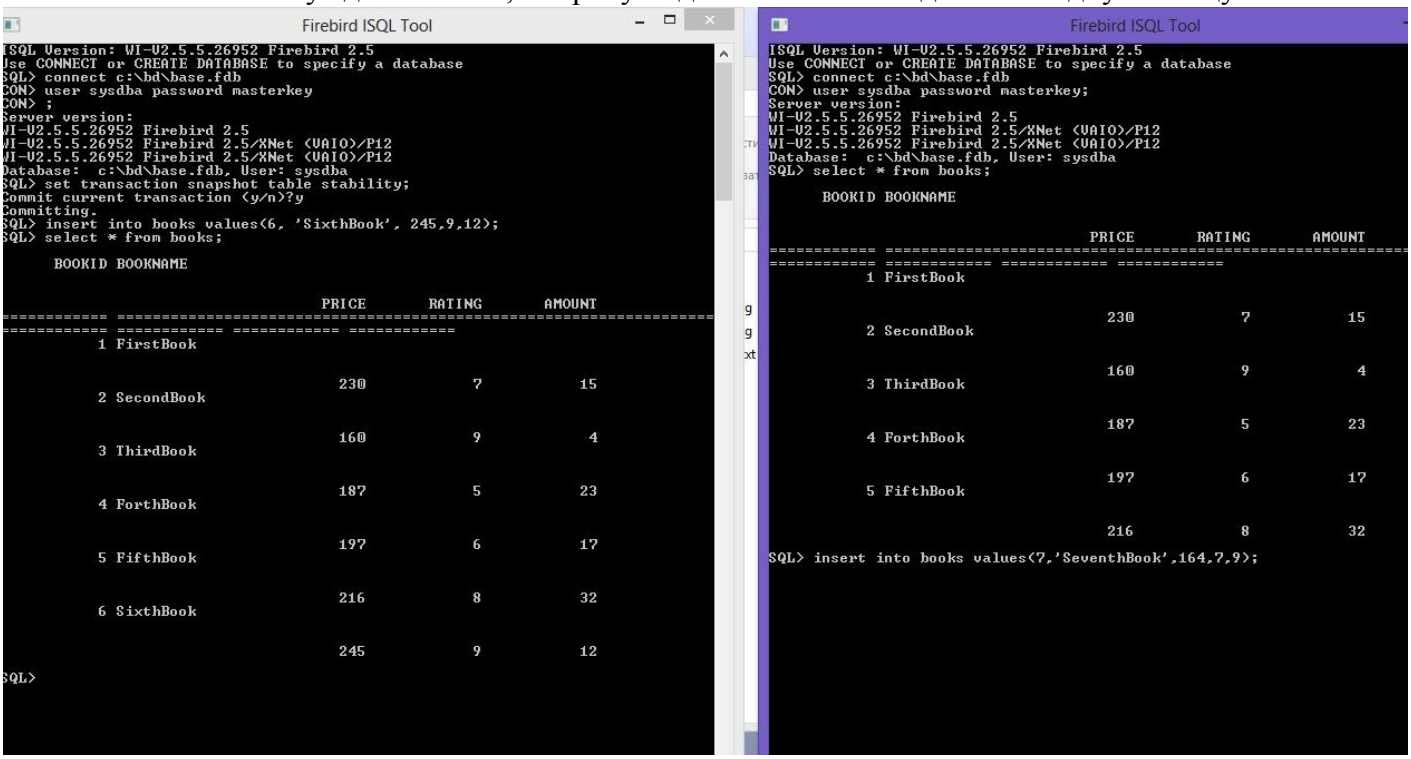


Рис.3. SNAPSHOT TABLE STABILITY: Попытка добавить данные из двух клиентов в одну таблицу

Как видно из рисунка 3, второй клиент не может добавить запись, в то время как у первого это удалось. Второй клиент ожидает завершения транзакции первого. Завершим её.

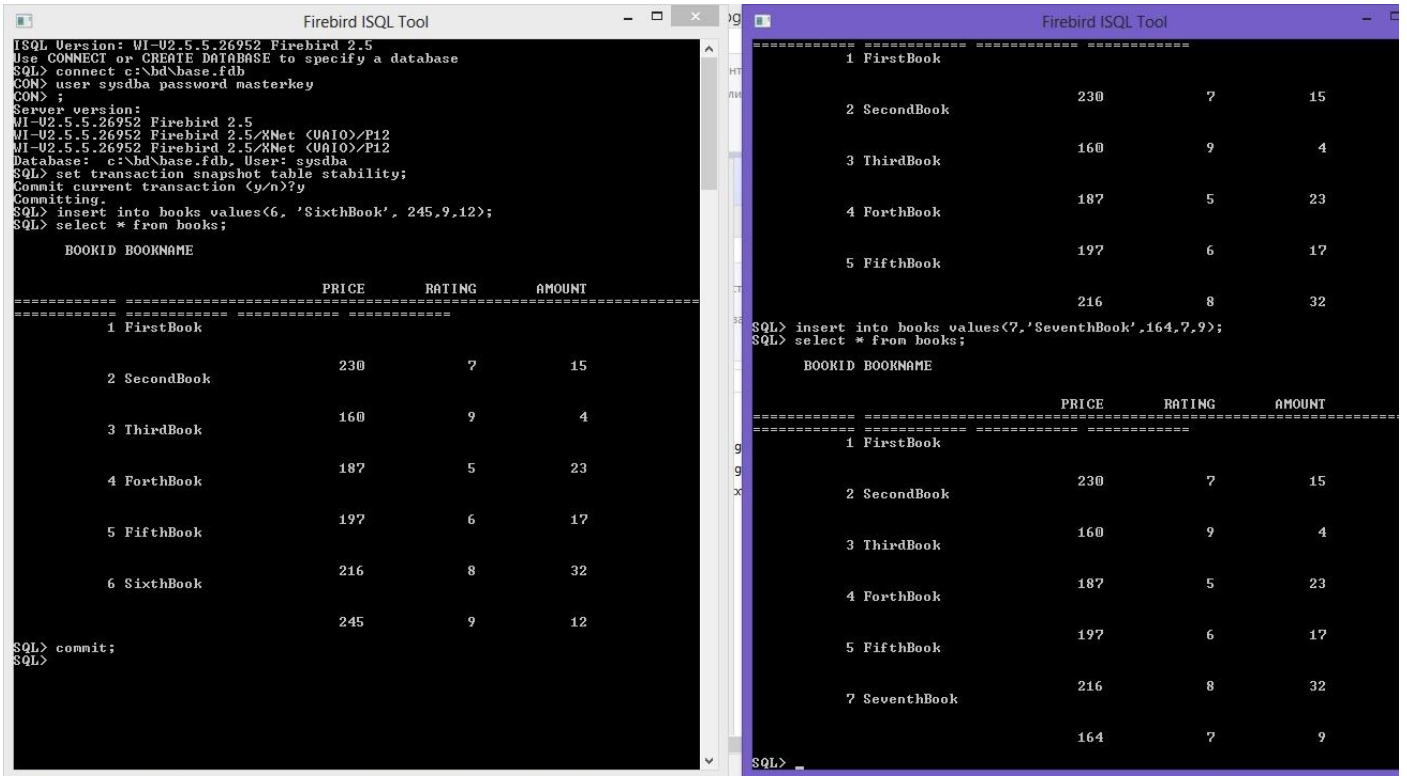


Рис.4. SNAPSHOT TABLE STABILITY: Второй клиент смог произвести запись.

Как видно из рисунка 4, второй клиент смог добавить запись в таблицу после завершения транзакции первого.

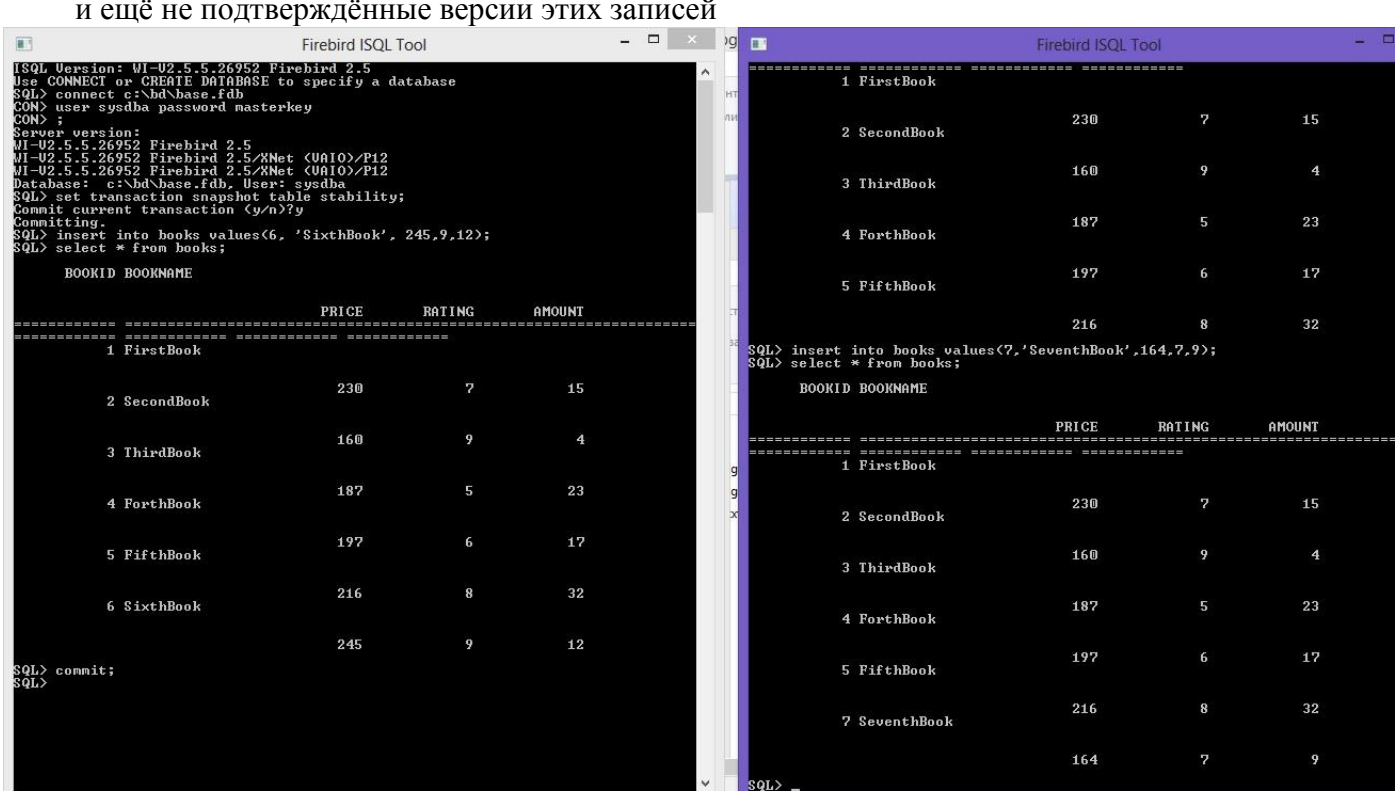
READ COMMITTED:

Уровень изолированности 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.

Для этого уровня изолированности можно указать один из двух значений дополнительной характеристики в зависимости от желаемого способа разрешения конфликтов: RECORD_VERSION и NO RECORD_VERSION.

RECORD_VERSION:

При задании RECORD_VERSION транзакция всегда читает последнюю подтверждённую версию записей таблиц, независимо от того, существуют ли изменённые и ещё не подтверждённые версии этих записей



.Рис.5. Первый клиент увидел изменения в таблице после коммита второго клиента.

Как видно из рисунка 5 первому клиенту необязательно заканчивать транзакцию для того чтобы увидеть изменения в таблице.

NO RECORD_VERSION (значение по умолчанию) В этом случае транзакция не может прочитать любую запись, которая была изменена параллельной активной (неподтвержденной) транзакцией.

Если указана стратегия разрешения блокировок WAIT, то это приведёт к ожиданию завершения или откату конкурирующей транзакции.

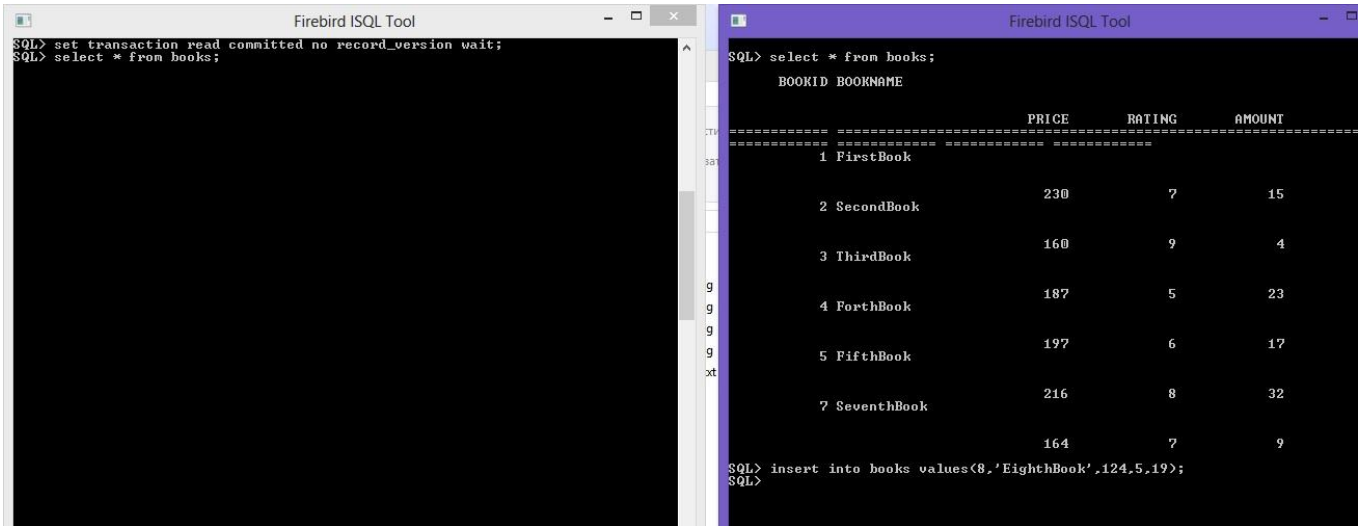


Рис.6. Первый клиент ожидает завершения транзакции второго, чтобы выполнить select.

Как видно из рисунка 6 первый клиент находится в состоянии ожидания завершения транзакции второго клиента.

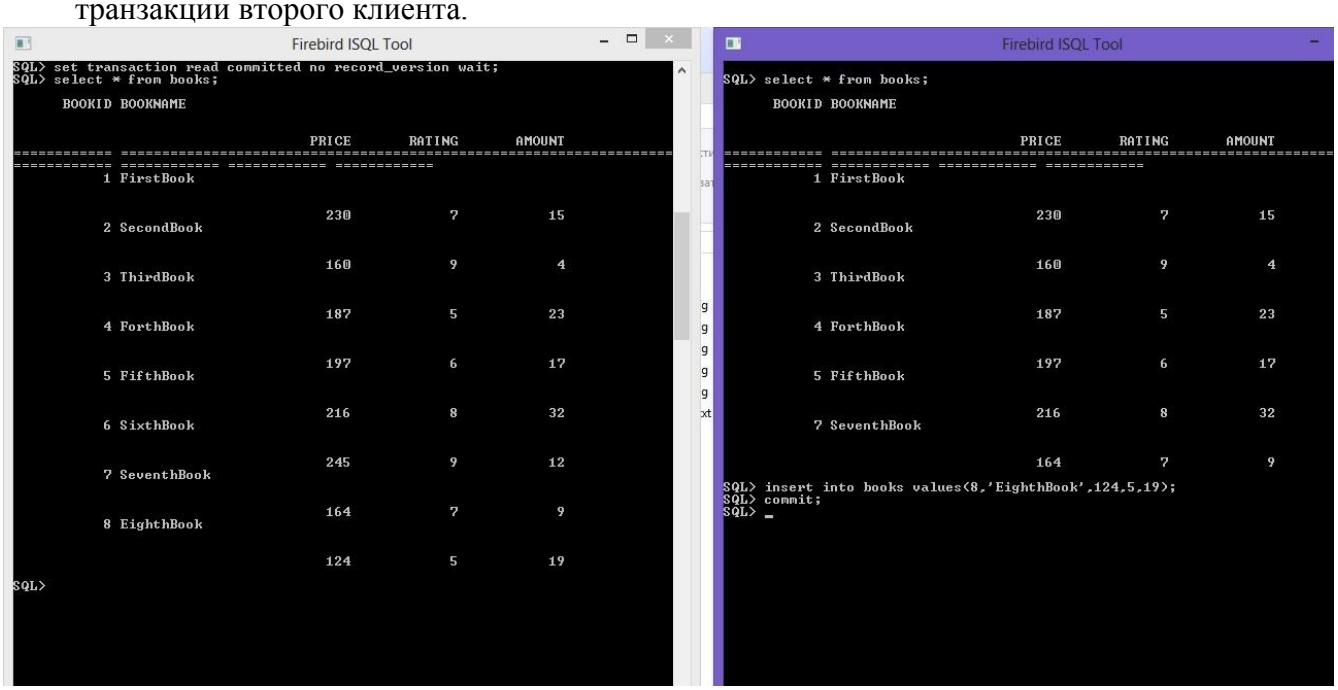


Рис.7. Первый клиент активен после коммита второго.

Как видно из рисунка 7, первый клиент сразу вывел обновленную таблицу после завершения транзакции второго клиента.

Если указана стратегия разрешения блокировок NO WAIT, то будет немедленно выдано соответствующее исключение.

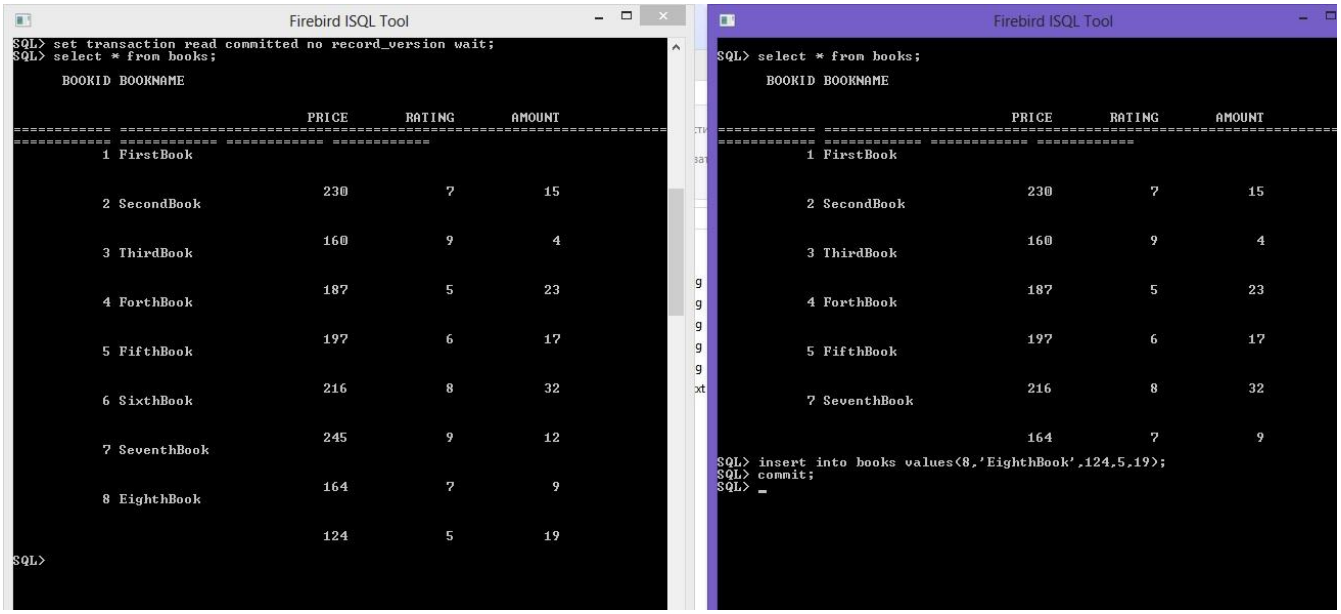


Рис.8. Режим read committed no record_version no wait.

Как видно из рисунка 8 первый клиент выдал исключение в связи с тем что второй клиент не закончил транзакцию.

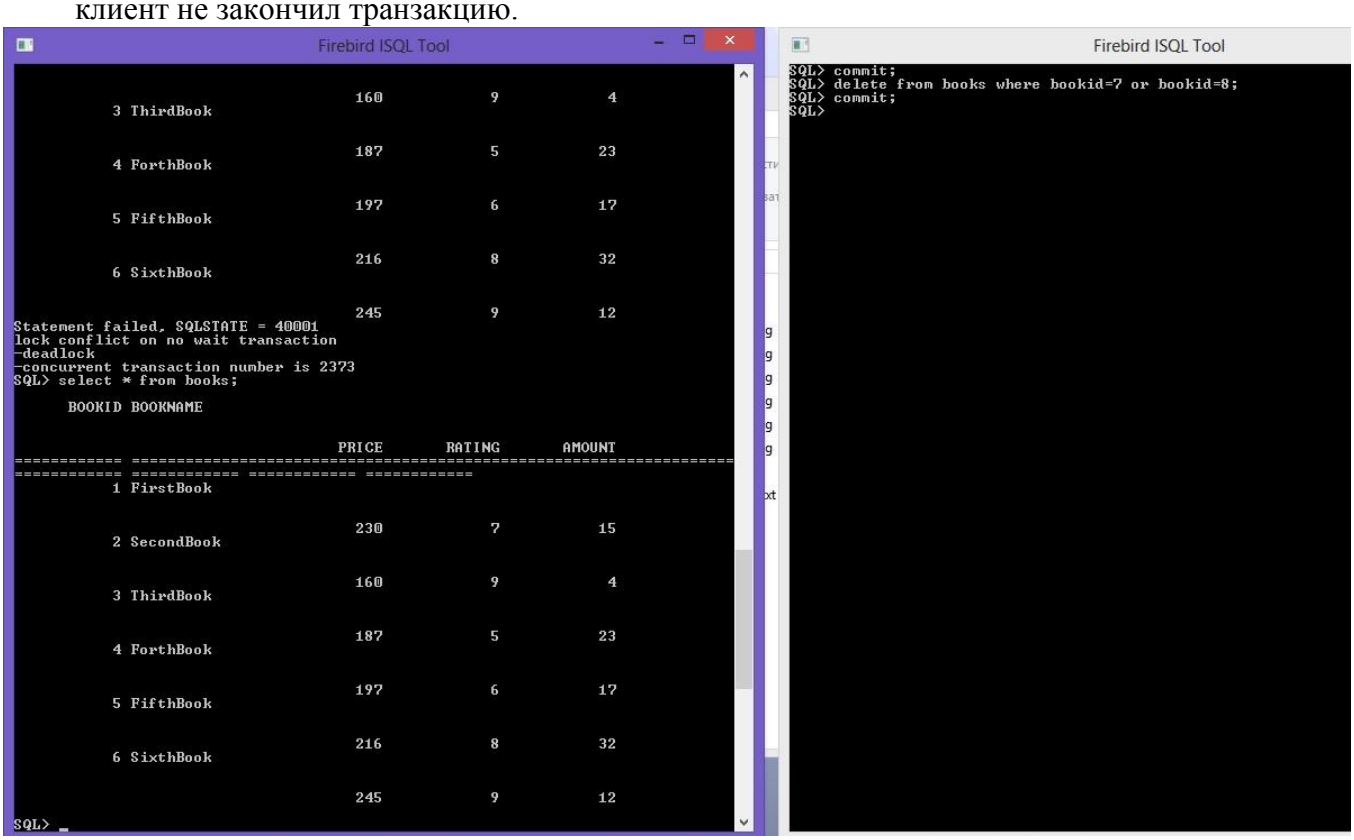


Рис.9. Режим read committed no record_version no wait.

После завершения транзакции второго клиента, первый клиент вывел обновленную таблицу.

Вывод

Транзакции нужны для обеспечения корректной работы нескольких пользователей с одной БД. Используя режимы транзакций, мы можем контролировать приоритетность пользователей. К примеру заставить ожидать всех пользователей пока один из них не изменит таблицу. В таком случае мы предполагаем что изменения наиболее важны. Либо же разрешить работу всех пользователей с какой-либо фиксированной версией БД, и обновить ее по окончании транзакции.

Уровни изолированности транзакций позволяют нам разгрузить систему. При уровне изоляции SNAPSHOT TABLE STABILITY мы запрещаем кому-либо изменять таблицу, с которой уже работает какой-либо пользователь. Это обеспечивает максимальную безопасность, но сильно нагружает систему. С другой стороны, остальные уровни изоляции менее строгие и разрешают параллельную работу с одной таблицей. Таким образом, мы можем выделить группы команд, по их приоритетности и назначить для них свои уровни изоляции, что позволит снизить нагрузку на систему.

Большинство действий с базами данных включает в себя несколько запросов внутри одной транзакции. Транзакция гарантирует, что все её запросы будут выполнены или не выполнены совсем. Простейшим примером важности транзакций является банковская система. При переводе средств с одного счета на другой необходимо совершить два действия: прибавить сумму на одном счете и вычесть на другом. Одно из действий может быть недоступно, тогда и второе не должно быть выполнено. Также это предотвратит изменения в БД при неожиданном обрыве канала связи с сервером. Не возникнет ситуации, когда выполнена только часть транзакции.