Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Базы данных

Отчет по лабораторной работе №7
Изучение работы транзакций

Работу выполнила:
Темнова А.С.
Группа: 43501/3
Преподаватель:
Мяснов А.В.

Санкт-Петербург
2016

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.

Работа проводится в IBExpert. Для проведения экспериментов параллельно запускается несколько сессий связи с БД, в каждой сессии настраивается уровень изоляции транзакций. Выполняются конкурентные операции чтения/изменения данных в различных сессиях, а том числе приводящие к конфликтам.

3 Ход выполнения работы

3.1 Изучить основные принципы работы транзакций.

Транзакция - это логический блок, объединяющий одну или более операций в базе данных и позволяющий подтвердить или отменить результаты работы всех операций в блоке. Иначе говоря, последовательность операций с базой данных, логически выполняемая как единое целое.

Транзакция обладает свойствами атомарности, согласованности, изоляции и долговременности (по-английски ACID - Atomicity, Consistency, Isolation, Durability).

1. Атомарность. Выполнение по принципу "все или ничего";
2. Согласованность. В результате транзакции система переходит из одного абстрактного корректного состояния в другое;
3. Изоляция. Данные, находящиеся в несогласованном состоянии, не должны быть видны другим транзакциям, пока изменения не будут завершены;
4. Долговременность. Если транзакция зафиксирована, то ее результаты должны быть долговечными. Новые состояния всех объектов сохранятся даже в случае аппаратных или системных сбоев.

Операции, о которых идет речь в определении - это INSERT/UPDATE/DELETE и SELECT. Если транзакция объединяет какие-то операции в единый блок, то говорят, что эти действия выполняются в контексте данной транзакции.

Транзакция обычно начинается автоматически при подключении клиента к БД и продолжается до выполнения команд COMMIT или ROLLBACK, либо до отключения клиента или сбоя сервера.

3.2 Провести эксперименты по запуску, подтверждению и откату транзакций

Рассмотрим команды COMMIT и ROLLBACK на примере добавления записей в таблицу.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  /* Create temp table */
4  create table my_wine (
5    my_wine_id      integer   not null,
6    wine_name   varchar(10) not null
7  );
8  alter table my_wine add constraint pk_my_wine primary key (my_wine_id);
9
10 /* Add some data without commit */
11 insert into my_wine values (1, 'Asti');
12 insert into my_wine values (2, 'Lambrusco');
```

```
13  insert into my_wine values (3, 'Chateaux');
14
15  /* Look */
16  select * from my_wine;
17
18  MY_WINE_ID WINE_NAME
19  ========== ==========
20  1 Asti
21  2 Lambrusco
22  3 Chateaux
23
24  /* Rollback */
25  rollback;
26  select * from my_wine;
27  /* Nothing! */
28
29  /* Add data again and commit */
30  insert into my_wine values (1, 'Asti');
31  insert into my_wine values (2, 'Lambrusco');
32  insert into my_wine values (3, 'Chateaux');
33  commit;
34
35  /* Look */
36  select * from my_wine;
37
38  MY_WINE_ID WINE_NAME
39  ========== ==========
40  1 Asti
41  2 Lambrusco
42  3 Chateaux
43
44  /* Rollback */
45  rollback;
46  select * from my_wine;
47
48  MY_WINE_ID WINE_NAME
49  ========== ==========
50  1 Asti
51  2 Lambrusco
52  3 Chateaux
53  /* Rollback doesn't work!!! */
54
55  /*****************************************************/
56
57  /*Save current state and delete record*/
58  savepoint first_state;
59  delete from my_wine where my_wine_id = 2;
60  select * from my_wine;
61
62  MY_WINE_ID WINE_NAME
63  ========== ==========
64  1 Asti
65  3 Chateaux
66
67  /*Insert record and save new state*/
68  insert into my_wine values (4, 'Amanti') ;
69  savepoint second_state;
70  select * from my_wine;
71
72  MY_WINE_ID WINE_NAME
73  ========== ==========
74  1 Asti
75  3 Chateaux
76  4 Amanti
77
78  /*Add new data to the table and rollback to the second state */
79  insert into my_wine values (5, 'Pomerol') ;
80  select * from my_wine;
81
82  MY_WINE_ID WINE_NAME
83  ========== ==========
84  1 Asti
85  3 Chateaux
86  4 Amantiliado
87  5 Pomerol
88
```

```
89  rollback to second_state;
90  select * from my_wine;
91
92  MY_WINE_ID WINE_NAME
93  ========== ==========
94  1 Asti
95  3 Chateaux
96  4 Amantiliado
97  /*We don't see inserted data*/
98
99  /*Rollback to the first state*/
100 rollback to first_state;
101 select * from my_wine;
102
103 MY_WINE_ID WINE_NAME
104 ========== ==========
105 1 Asti
106 2 Lambrusco
107 3 Chateaux
108 /*Now we see deleted data 2*/
109
110 commit;
```

Листинг 1: rollback и commit

3.3 Разобраться с уровнями изоляции транзакций в Firebird

Firebird предоставляет три уровня изоляции транзакций для определения "глубины"согласованности требований транзакции. В одном крайнем случае транзакция может получить исключительный доступ по записи ко всей таблице, в то время как в другом крайнем случае неподтвержденная транзакция получает доступ к любым внешним изменениям состояния базы данных. Никакая транзакция в Firebird не сможет видеть неподтвержденные изменения данных от других транзакций.

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. Рассмотрим уровни изоляции более подробно.

1. READ COMMITTED. Самый низкий уровень изоляции. Буквально переводится как "читать подтвержденные данные".

   Только на этом уровне изоляции вид состояния базы данных, измененного в процессе выполнения транзакции, изменяется каждый раз, когда подтверждаются версии записей. Вновь подтвержденная версия записи заменяет ту версию, которая была видна при старте транзакции. Подтвержденные добавления, выполненные после старта транзакции, становятся видимыми для нее.

2. SNAPSHOT. "Средний"уровень изоляции.

   SNAPSHOT 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 подтвержденных (и разумеется, неподтвержденных) транзакциях, не видны в этой транзакции. В то же время SNAPSHOT не блокирует данные, которые он не изменяет.

3. SNAPSHOT TABLE STABILITY. Самый "глубокий"уровень изоляции.

   Блокировка на уровне таблицы, создаваемая этим уровнем изоляции, включает в себя все таблицы, к которым осуществляет доступ транзакция, включая те, которые связаны ссылочными ограничениями.

   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.

3.4 Спланировать и провести эксперименты, показывающие основные возможности транзакций с различным уровнем изоляции.

3.4.1 READ COMMITTED транзакция

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 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.

1.1. При установке RECORD_VERSION сервер позволяет транзакции читать самую последнюю подтвержденную версию записи. Если транзакция имеет режим READ WRITE (чтение/запись), то она будет способна перезаписывать самую последнюю подтвержденную версию записи, если ее идентификатор (TID) более поздний, чем идентификатор транзакции, подтвердившей самую последнюю версию записи.

1.2. При установке NO_RECORD_VERSION (по умолчанию) сервер эффективно имитирует поведение систем, которые используют двухфазную блокировку для управления параллельностью. Он блокирует для текущей транзакции чтение строки, если для нее существует изменение, ожидающее завершения. Разрешение ситуации зависит от установки разрешения блокировки.

2.1. При задании WAIT (по умолчанию) транзакция будет ждать, когда другая транзакция либо подтвердит, либо отменит свои изменения. Эти изменения затем станут доступными, если другая транзакция их отменит или если идентификатор транзакции более поздний, чем идентификатор другой транзакции. Транзакция будет завершена аварийно по конфликту блокировки, если идентификатор другой транзакции будет более поздним.

2.2. При задании NOWAIT транзакция немедленно получит сообщение о конфликте блокировки.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  select * from my_wine;
4
5  MY_WINE_ID WINE_NAME
6  ========== ==========
7  1 Asti
8  2 Lambrusco
9  3 Chateaux
10
11 insert into my_wine values (4, 'Bordo') ;
12 select * from my_wine;
13
14 MY_WINE_ID WINE_NAME
15 ========== ==========
16 1 Asti
17 2 Lambrusco
18 3 Chateaux
19 4 Bordo
20
21 commit;
```

Листинг 2: READ COMMITTED транзакция (терминал 1)

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  set transaction isolation level read committed;
4  Commit current transaction (y/n)?n
5  Rolling back work.
6
7  select * from my_wine;
8
9  MY_WINE_ID WINE_NAME
```

```
10  ============ ==========
11  1 Asti
12  2 Lambrusco
13  3 Chateaux
14
15  select * from my_wine;
16
17  MY_WINE_ID WINE_NAME
18  ============ ==========
19  1 Asti
20  2 Lambrusco
21  3 Chateaux
22  4 Bordo
```

Листинг 3: READ COMMITTED транзакция (терминал 2)

Выводим таблицу данных в обоих терминалах. Всё совпадает (см.рис.1). В терминале 1 вносим изменения в таблице (добавляем запись). Во 2 терминале при запросе на вывод таблицы транзакция переходит в режим ожидания (см.рис.2). После подтверждения изменений в таблице в 1 терминале во 2 выведены корректные данные (см.рис.3).

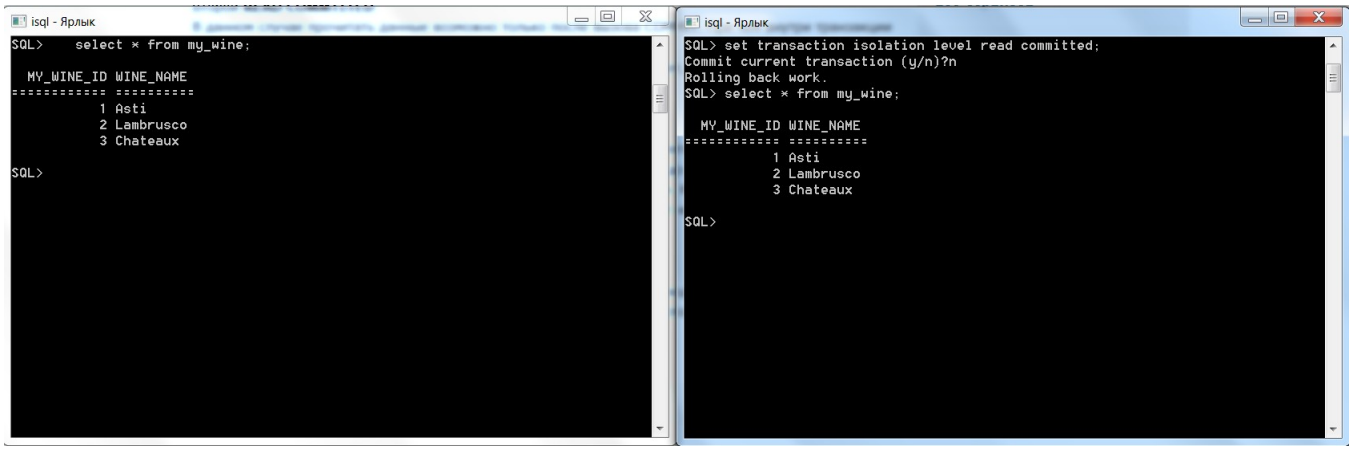

Рис. 1: READ COMMITTED транзакция, шаг 1

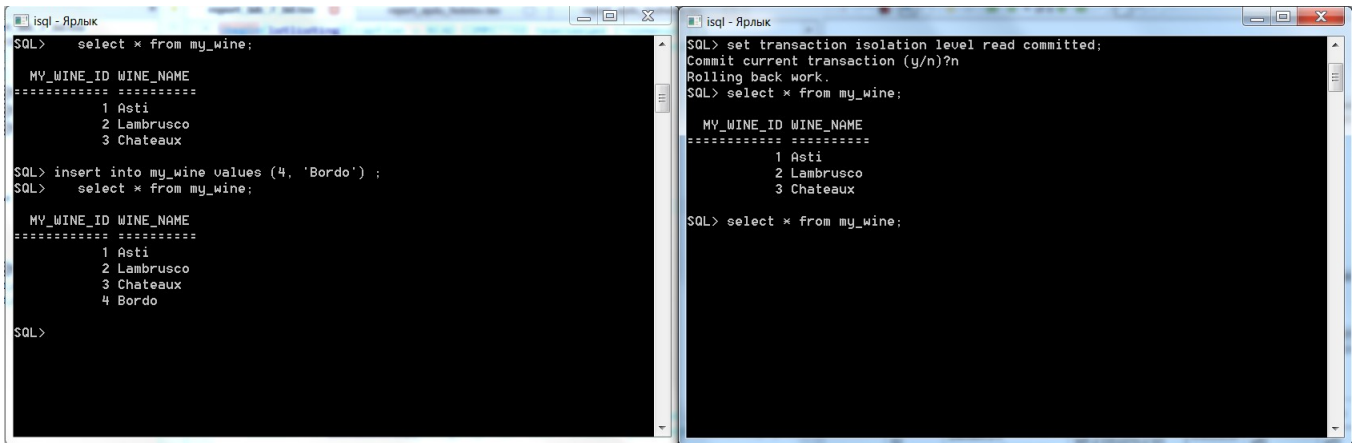

Рис. 2: READ COMMITTED транзакция, шаг 2

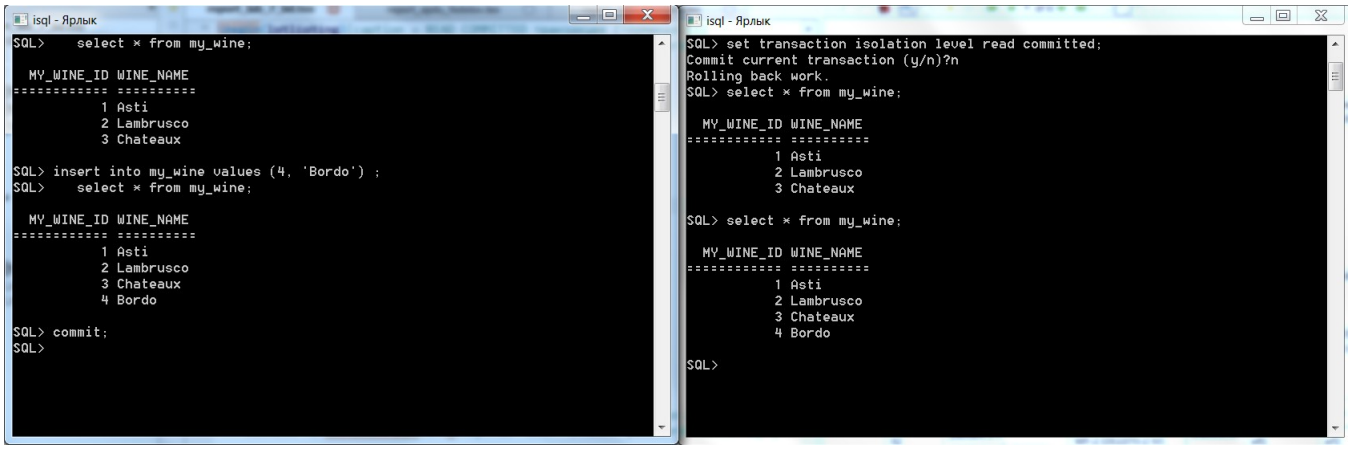

Рис. 3: READ COMMITTED транзакция, шаг 3

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

```
1   update my_wine set wine_name = 'Bordo' where wine_name = 'Amanti';
2
3   select * from my_wine;
4
5   MY_WINE_ID WINE_NAME
6   ========== ==========
7   1 Asti
8   2 Lambrusco
9   3 Chateaux
10  4 Bordo
11
12  commit;
```

Листинг 4: READ COMMITTED NO WAIT транзакция (терминал 1)

```
1   set transaction isolation level read committed no wait;
2   Commit current transaction (y/n)?n
3   Rolling back work.
4
5   select * from my_wine;
6
7   MY_WINE_ID WINE_NAME
8   ========== ==========
9   1 Asti
10  2 Lambrusco
11  3 Chateaux
12  4 Amanti
13
14  select * from my_wine;
15
16  MY_WINE_ID WINE_NAME
17  ========== ==========
18  1 Asti
19  2 Lambrusco
20  3 Chateaux
21  Statement failed, SQLSTATE = 40001
22  lock conflict on no wait transaction
23  -deadlock
24  -concurrent transaction number is 3096
25
26  select * from my_wine;
27
28  MY_WINE_ID WINE_NAME
29  ========== ==========
30  1 Asti
31  2 Lambrusco
32  3 Chateaux
33  4 Bordo
```

Листинг 5: READ COMMITTED NO WAIT транзакция (терминал 2)

В терминале 1 вносим изменения в таблице (см.рис.4). Во 2 терминале при запросе на вывод таблицы немедленно выводится исключение (см.рис.5) - ошибка, т.к. другая транзакция изменила таблицу и не подтвердила изменения.

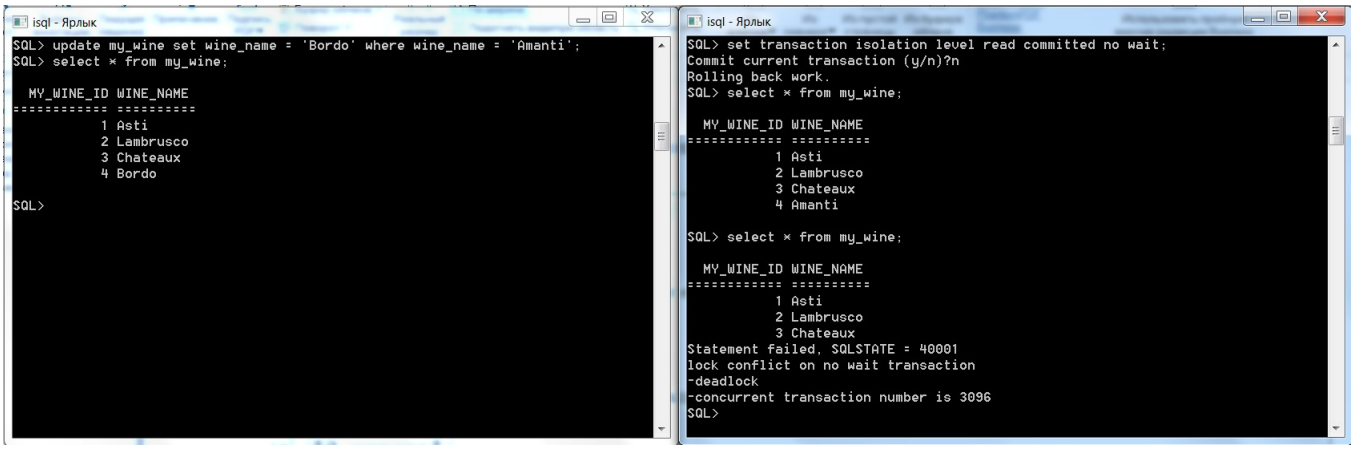

Рис. 4: READ COMMITTED NO WAIT транзакция, шаг 1

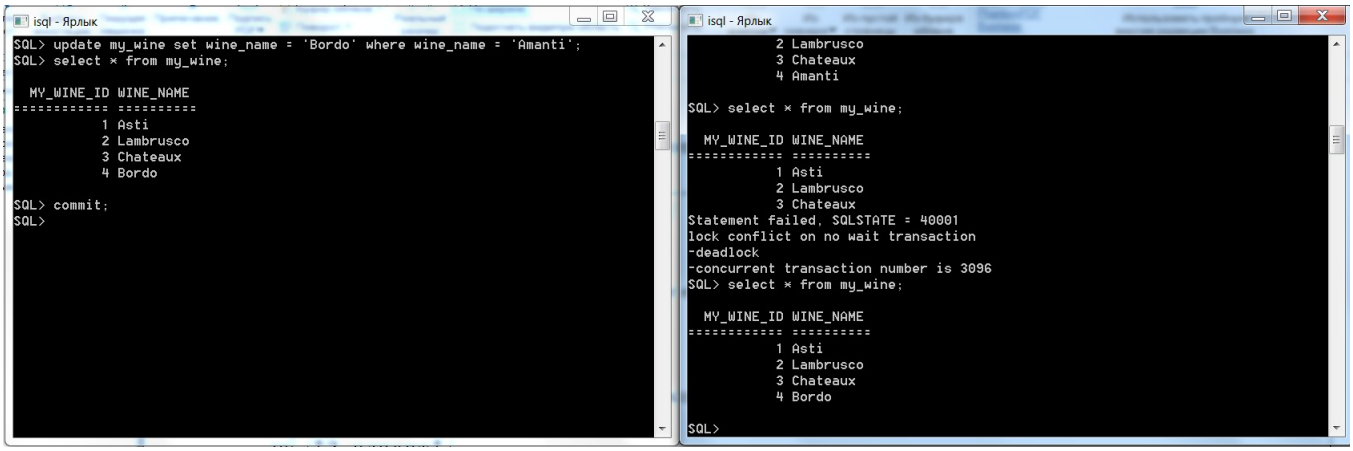

Рис. 5: READ COMMITTED NO WAIT транзакция, шаг 2

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.

```
1   select * from my_wine;
2
3   MY_WINE_ID WINE_NAME
4   ========== ==========
5   1 Asti
6   2 Lambrusco
7   3 Chateaux
8   4 Bordo
9   5 Clamise
10
11  insert into my_wine values (6, 'Isabel');
12  select * from my_wine;
13
14  MY_WINE_ID WINE_NAME
15  ========== ==========
16  1 Asti
17  2 Lambrusco
18  3 Chateaux
19  4 Bordo
20  5 Clamise
21  6 Isabel
22
```

```
23  commit;
24  select * from my_wine;
25
26  MY_WINE_ID WINE_NAME
27  ========== ==========
28  1 Asti
29  2 Lambrusco
30  3 Chateaux
31  4 Bordo
32  5 Clamise
33  6 Isabel
```

Листинг 6: READ COMMITTED RECORD_VERSION транзакция (терминал 1)

```
1   set transaction isolation level read committed record_version;
2   Commit current transaction (y/n)?n
3   Rolling back work.
4
5   select * from my_wine;
6
7   MY_WINE_ID WINE_NAME
8   ========== ==========
9   1 Asti
10  2 Lambrusco
11  3 Chateaux
12  4 Bordo
13  5 Clamise
14
15  select * from my_wine;
16
17  MY_WINE_ID WINE_NAME
18  ========== ==========
19  1 Asti
20  2 Lambrusco
21  3 Chateaux
22  4 Bordo
23  5 Clamise
24
25  select * from my_wine;
26
27  MY_WINE_ID WINE_NAME
28  ========== ==========
29  1 Asti
30  2 Lambrusco
31  3 Chateaux
32  4 Bordo
33  5 Clamise
34  6 Isabel
```

Листинг 7: READ COMMITTED RECORD_VERSION транзакция (терминал 2)

Проверяем, что данные совпадают, и в 1 терминале вносим изменения в таблицу (см.рис.6). Опять выводит таблицы - данные различны. В терминале 1 измененные, в терминале 2 те же, что и были (см.рис.7). Затем в терминале 1 подтверждаем (commit) внесенные изменения и снова в обоих терминалах выводим данные. Всё совпало (см.рис.8).

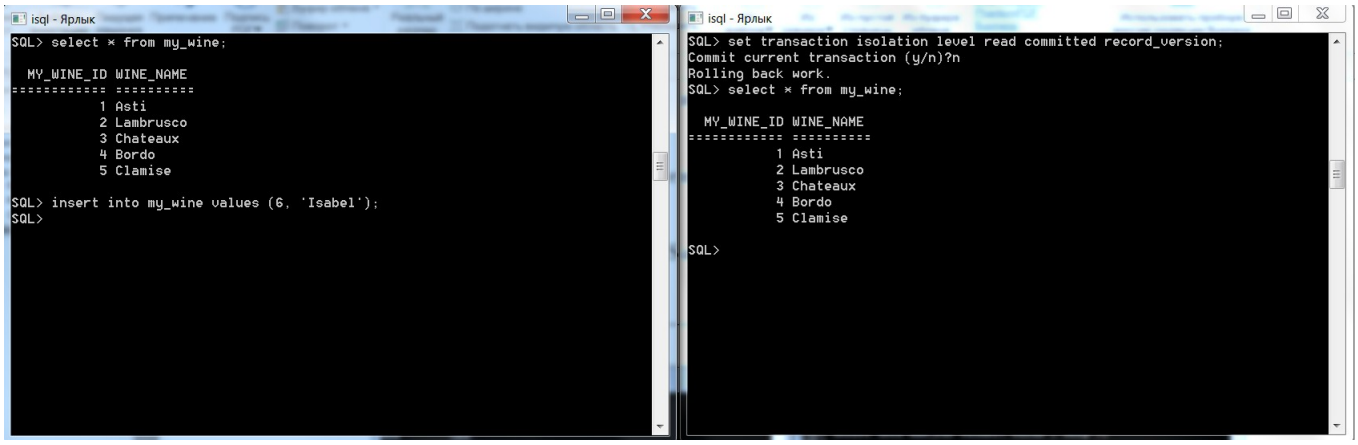

Рис. 6: READ COMMITTED RECORD_VERSION транзакция, шаг 1

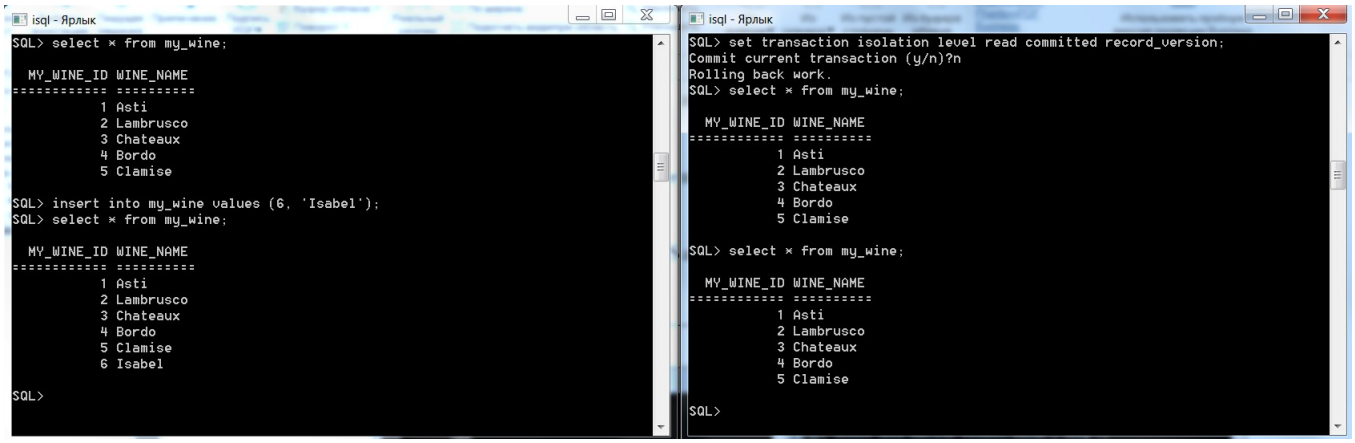

Рис. 7: READ COMMITTED RECORD_VERSION транзакция, шаг 2

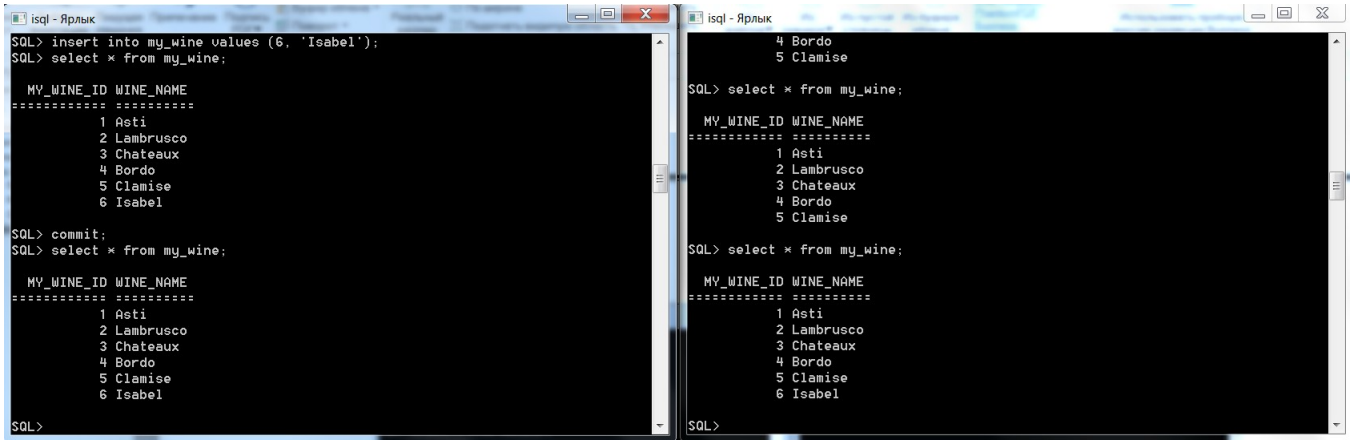

Рис. 8: READ COMMITTED RECORD_VERSION транзакция, шаг 3

3.4.2 SNAPSHOT транзакция

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.

```
connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';

select * from my_wine;

MY_WINE_ID WINE_NAME
========== ==========
1 Asti
2 Lambrusco
3 Chateaux
4 Bordo

insert into my_wine values (5, 'Clamise');
commit;
select * from my_wine;

MY_WINE_ID WINE_NAME
========== ==========
1 Asti
2 Lambrusco
3 Chateaux
4 Amanti

commit;
```

Листинг 8: SNAPSHOT транзакция (терминал 1)

```
connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';

set transaction isolation level snapshot;
Commit current transaction (y/n)?n
Rolling back work.

select * from my_wine;

MY_WINE_ID WINE_NAME
========== ==========
1 Asti
2 Lambrusco
3 Chateaux
4 Bordo

select * from my_wine;

MY_WINE_ID WINE_NAME
========== ==========
1 Asti
2 Lambrusco
3 Chateaux
4 Bordo

commit;

set transaction isolation level snapshot;
Commit current transaction (y/n)?n
Rolling back work.

select * from my_wine;

MY_WINE_ID WINE_NAME
========== ==========
1 Asti
2 Lambrusco
3 Chateaux
4 Bordo
5 Clamise
```

Листинг 9: SNAPSHOT транзакция (терминал 2)

Запустили транзакцию. Сравнили таблицы в двух терминалах. Совпадают. В терминале 1 внесли изменения в таблицу. В терминале 2 эти изменения не видны (рис.9). После завершения транзакции в терминале 2 и повторном ее запуске изменения, внесенные в терминале 1, становятся видны (рис.10).

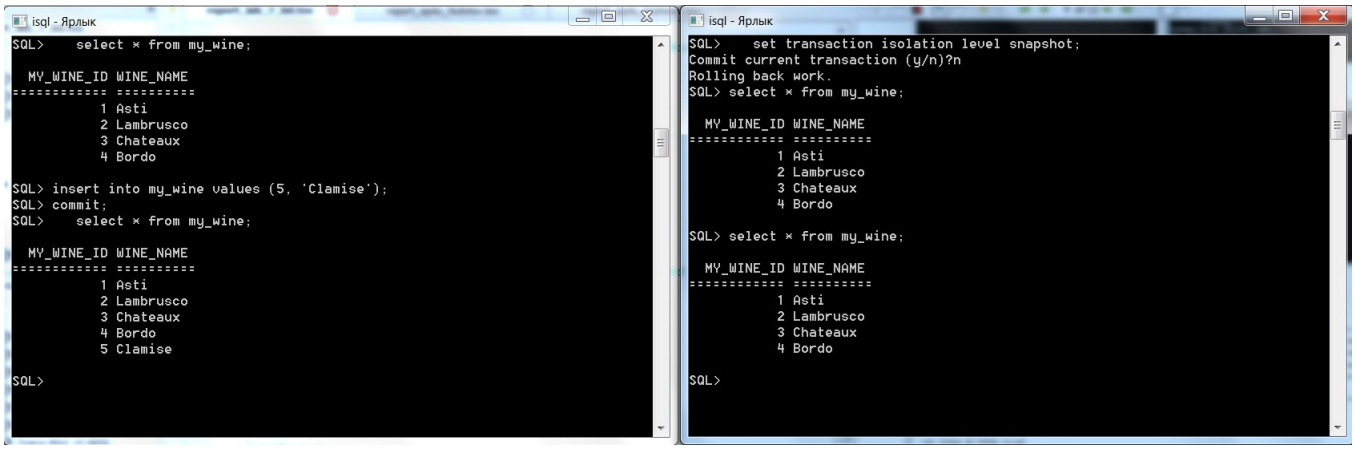

Рис. 9: SNAPSHOT транзакция, шаг 1

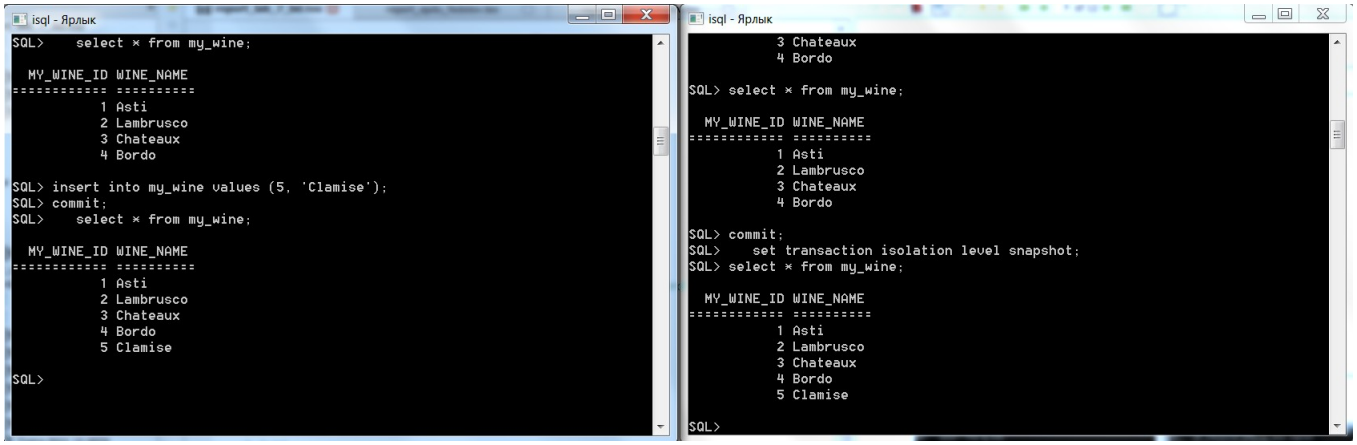

Рис. 10: SNAPSHOT транзакция, шаг 2

3.4.3 SNAPSHOT TABLE STABILITY транзакция

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.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  set transaction isolation level snapshot table stability;
4  Commit current transaction (y/n)?n
5  Rolling back work.
6
7  insert into my_wine values (7, 'Merlot');
8  select * from my_wine;
9
10 MY_WINE_ID WINE_NAME
11 ============ ==========
12 1 Asti
13 2 Lambrusco
14 3 Chateaux
15 4 Bordo
16 5 Clamise
17 6 Isabel
18 7 Merlot
19
20 commit;
```

Листинг 10: SNAPSHOT TABLE STABILITY транзакция (терминал 1)

```
1   connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3   set transaction isolation level snapshot table stability;
4   Commit current transaction (y/n)?n
5   Rolling back work.
6
7   select * from my_wine;
8
9   MY_WINE_ID WINE_NAME
10  ========== ==========
11  1 Asti
12  2 Lambrusco
13  3 Chateaux
14  4 Bordo
15  5 Clamise
16  6 Isabel
```

Листинг 11: SNAPSHOT TABLE STABILITY транзакция (терминал 2)

В терминале 1 внесли изменения в таблицу (рис.11). В терминале 2 ввели запрос на вывод таблицы, измененной в терминале 1, и транзакция перешла в режим ожидания (рис.12). После завершения транзакции в терминале 1 в терминале 2 происходит выполнение запроса (рис.13).

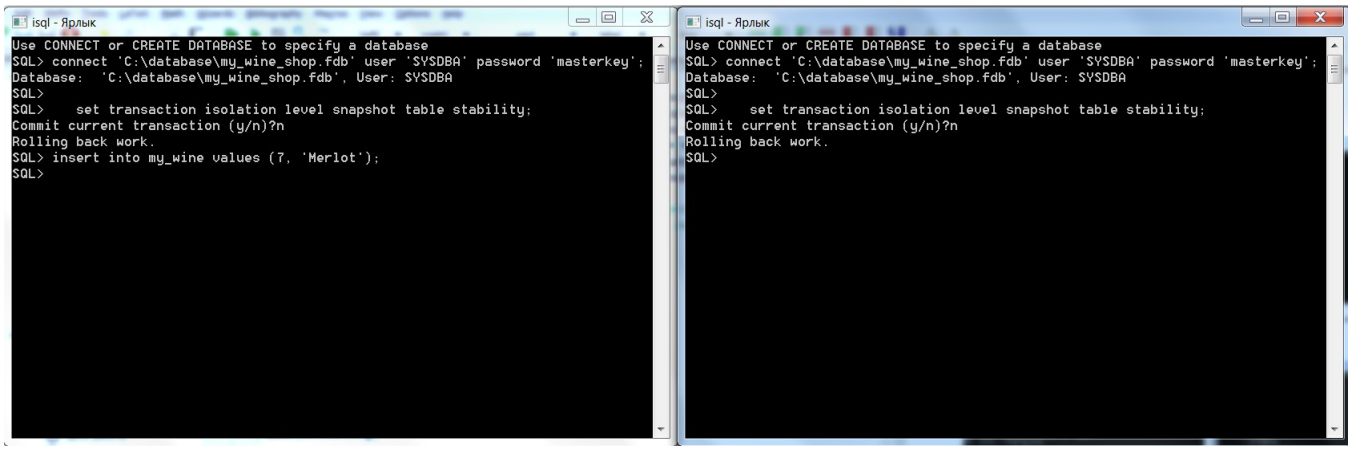

Рис. 11: SNAPSHOT TABLE STABILITY транзакциz, шаг 1

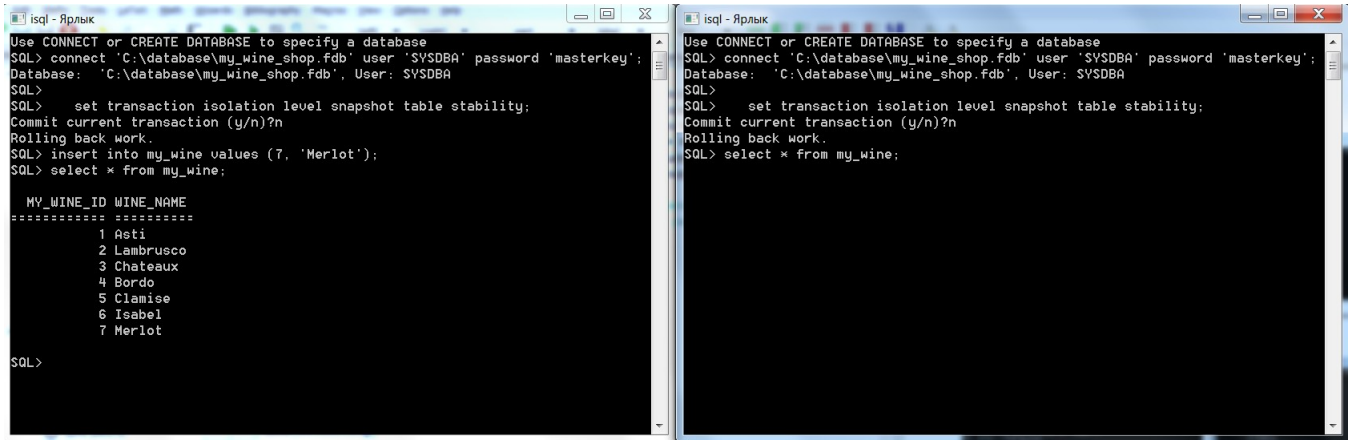

Рис. 12: SNAPSHOT TABLE STABILITY транзакциz, шаг 2

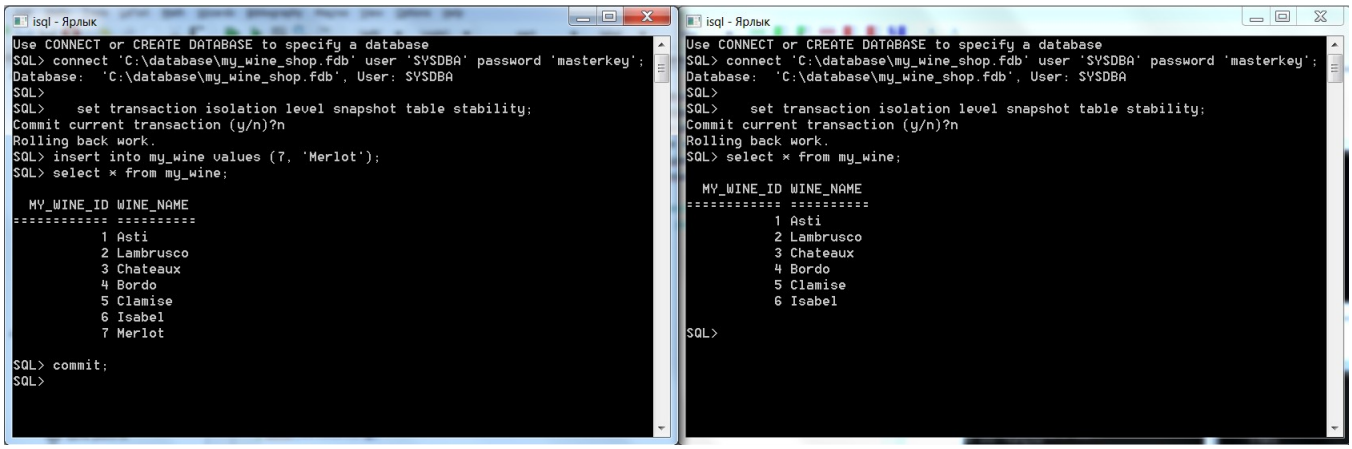

Рис. 13: SNAPSHOT TABLE STABILITY транзакциz, шаг 3

4 Выводы

В ходе данной работы я познакомилась с различными уровнями изоляции транзакций в Firbird.

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

Уровни изоляции транзакций позволяют разгрузить систему. При уровне изоляции SNAPSHOT TABLE STABILITY запрещается 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. Не возникнет ситуации, когда выполнена только часть транзакции.