

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»

SQL-программирование: Изучение работы транзакций

Выполнил

студент гр. 43501/3

Е.А. Никитин

Преподаватель

А.В. Мяснов

«__»_____2015г.

Санкт-Петербург

2015

Цели работы

Познакомить студентов 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:

1) 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. Транзакции обрабатываются транзакционными системами, в процессе работы которых создаётся история транзакций.

Существует два типа транзакций:

- Неявная транзакция - задает любую отдельную инструкцию INSERT, UPDATE или DELETE как единицу транзакции.
- Явная транзакция - обычно это группа инструкций языка Transact-SQL, начало и конец которой обозначаются такими инструкциями, как BEGIN TRANSACTION, COMMIT и ROLLBACK.

Свойства транзакций

Транзакции обладают следующими свойствами, которые все вместе обозначаются сокращением ACID (Atomicity, Consistency, Isolation, Durability):

- атомарность (Atomicity);
- согласованность (Consistency);
- изолированность (Isolation);
- долговечность (Durability).

Свойство атомарности обеспечивает неделимость набора инструкций, который модифицирует данные в базе данных и является частью транзакции. Это означает, что или выполняются все изменения данных в транзакции, или в случае любой ошибки осуществляется откат всех выполненных изменений.

Свойство согласованности обеспечивает, что в результате выполнения транзакции база данных не будет содержать несогласованных данных. Иными словами, выполняемые транзакцией трансформации данных переводят базу данных из одного согласованного состояния в другое.

Свойство изолированности отделяет все параллельные транзакции друг от друга. Иными словами, активная транзакция не может видеть модификации данных в параллельной или незавершенной транзакции. Это означает, что для обеспечения изоляции для некоторых транзакций может потребоваться выполнить откат.

Свойство долговечности обеспечивает одно из наиболее важных требований баз данных: сохраняемость данных. Иными словами, эффект транзакции должен оставаться действенным даже в случае системной ошибки. По этой причине, если в процессе выполнения транзакции происходит системная ошибка, то осуществляется откат для всех выполненных инструкций этой транзакции.

2) Провести эксперименты по запуску, подтверждению и откату транзакций.

Была создана база данных bd7, создана таблица station и заполнена данными:

```
SQL> insert into station(st_id, name) values(1, 'MOSCOW');
```

```
SQL> commit;
```

```
SQL> select * from station;
```

```
      ST_ID NAME
=====
=====

1      1 MOSCOW
```

```
SQL> insert into station(st_id, name) values(2, 'SPB');
```

```
SQL> insert into station(st_id, name) values(3, 'N_Novgorod');
```

```
SQL> insert into station(st_id, name) values(4, 'Vladimir');
```

```
SQL> insert into station(st_id, name) values(5, 'Kazan');
```

```
SQL> commit;
```

```
SQL> select * from station;
```

ST_ID	NAME
-------	------

1	MOSCOW
---	--------

2	SPB
---	-----

3	N_Novgorod
---	------------

4	Vladimir
---	----------

5	Kazan
---	-------

Добавим еще одну запись, создадим точку сохранения, удалим новую запись и вернемся к версии в точке сохранения:

```
SQL> insert into station(st_id, name) values(6, 'Vologda');
```

```
SQL> savepoint vologda_insr;
```

```
SQL> delete from station where st_id=6;
```

```
SQL> select * from station;
```

ST_ID	NAME
-------	------

1	MOSCOW
---	--------

2	SPB
---	-----

3	N_Novgorod
---	------------

4	Vladimir
---	----------

5	Kazan
---	-------

```
SQL> rollback to vologda_insr;
```

```
SQL> select * from station;
```

ST_ID	NAME
-------	------

1	MOSCOW
2	SPB
3	N_Novgorod
4	Vladimir
5	Kazan
6	Vologda

Как видно, после возврата к точке сохранения, запись о станции Vologda присутствует в таблице.

Теперь вернемся к моменту, когда эта запись еще не была создана:

```
SQL> rollback;
```

```
SQL> select * from station;
```

ST_ID	NAME
-------	------

1	MOSCOW
2	SPB
3	N_Novgorod
4	Vladimir

2) Эксперименты с разными уровнями изоляции.

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.

В Firebird уровень изоляции может быть:

* READ COMMITTED:

- RECORD_VERSION;
- NO RECORD_VERSION;

* SNAPSHOT;

* SNAPSHOT TABLE STABILITY.

READ COMMITTED

Самым низким уровнем изоляции является READ COMMITTED. 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.

SNAPSHOT (параллельность)

"Средним" уровнем изоляции является SNAPSHOT, альтернативно называемый Repeatable Read (повторяемое чтение) или concurrency (параллельность). При этом изоляция SNAPSHOT в Firebird не является в точности соответствующей уровню Repeatable Read, как он определен в стандарте. Он изолирует вид базы данных для транзакции изменениями на уровне строк только для существующих строк. При этом, поскольку по своей природе многоверсионная архитектура полностью изолирует транзакции

SNAPSHOT от новых строк, подтвержденных другими транзакциями, не предоставляя транзакциям SNAPSHOT доступ к глобальному образу состояния транзакций (TSB, см. главу 25), транзакции SNAPSHOT не могут видеть фантомные строки.

SNAPSHOT TABLE STABILITY (согласованность)

Самым "глубоким" уровнем изоляции является SNAPSHOT TABLE STABILITY, альтернативно называемый consistency, потому что он гарантирует получение данных в неизменном состоянии, который останется внешне согласованным в пределах базы данных, пока продолжается транзакция. Транзакции чтения/записи не могут даже читать таблицы, которые блокируются транзакцией с таким уровнем изоляции.

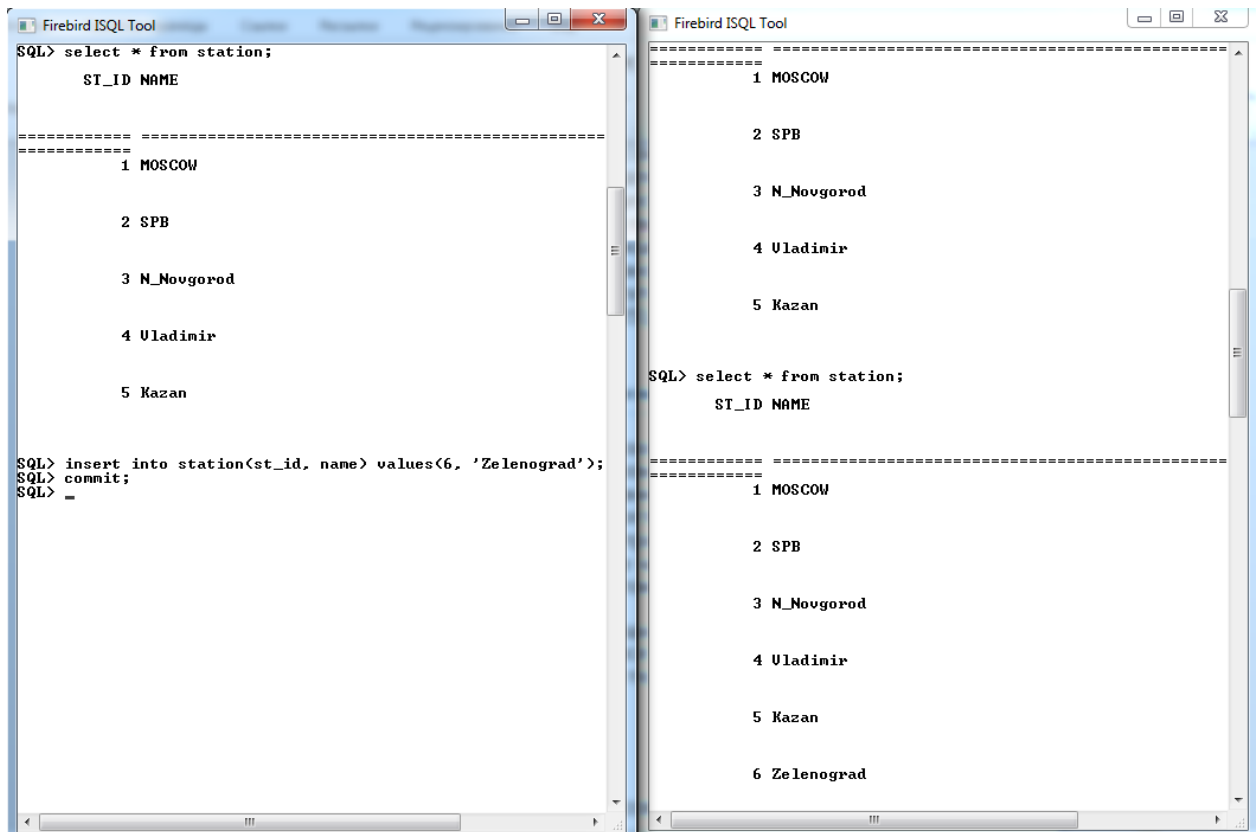
Блокировка на уровне таблицы, создаваемая этим уровнем изоляции, включает в себя все таблицы, к которым осуществляет доступ транзакция, включая те, которые связаны ссылочными ограничениями.

Эксперименты.

Уровень **READ COMMITTED**

А) Способ разрешения конфликтов: **RECORD_VERSION**

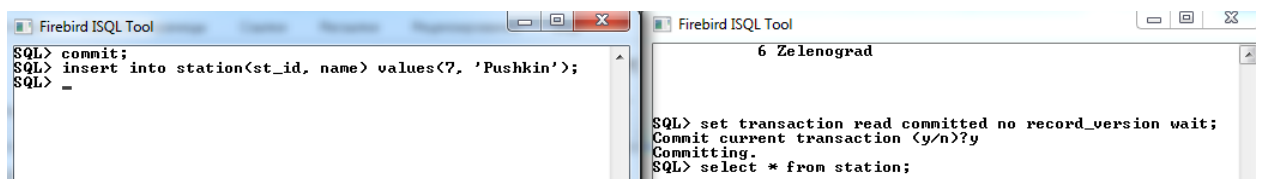
При задании **RECORD_VERSION** транзакция всегда читает последнюю подтверждённую версию записей таблиц, независимо от того, существуют ли изменённые и ещё не подтверждённые версии этих записей



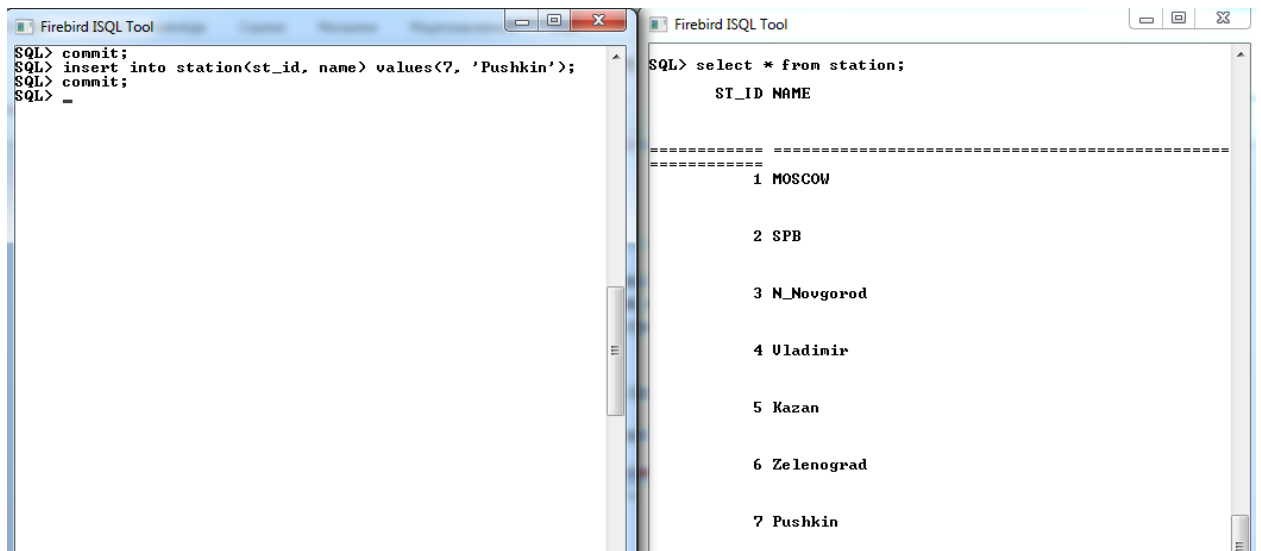
До коммита первого клиента (слева) содержимое таблицы соответствовало последней подтвержденной версии, после коммита второй клиент видит измененное содержимое таблицы сразу. Режим разрешения блокировок не влияет на поведение транзакции.

Б) Способ разрешения конфликтов: NO RECORD_VERSION

-режим разрешения блокировок wait:

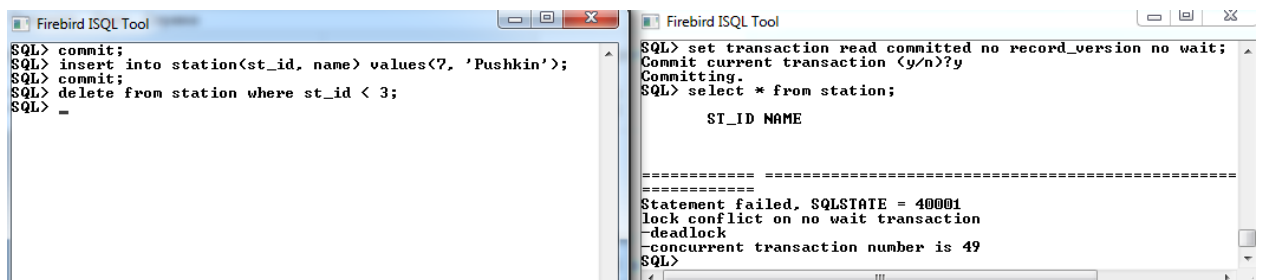


Пока первый клиент (слева) не закоммитил изменения, второй клиент не может посмотреть данные и ждет подтверждения.

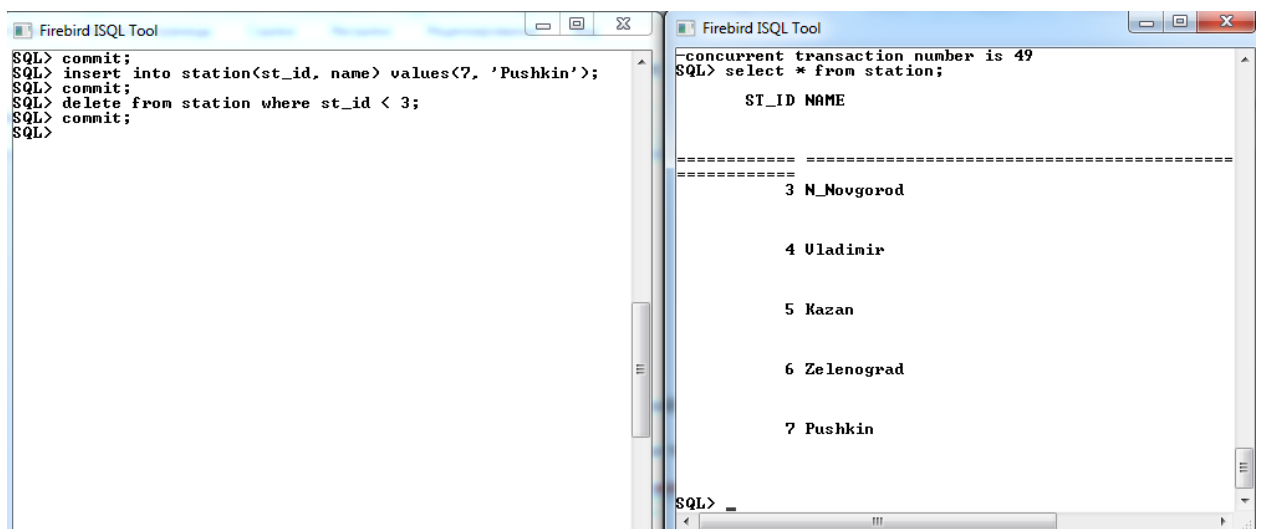


Первый клиент подтвердил и второй видит изменения .

- режим разрешения блокировок по wait



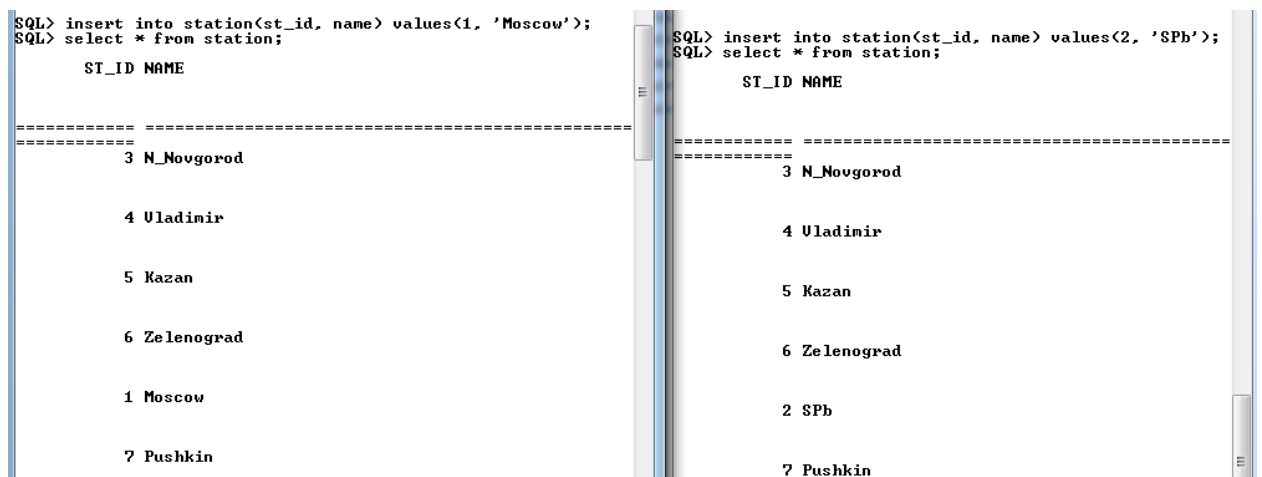
В этом режиме при попытке просмотра не подтвержденных данных выходит сообщение об ошибке – второй клиент не может посмотреть не предыдущую версию, не текущую (неподтвержденную).



После коммита все нормально.

Уровень 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.

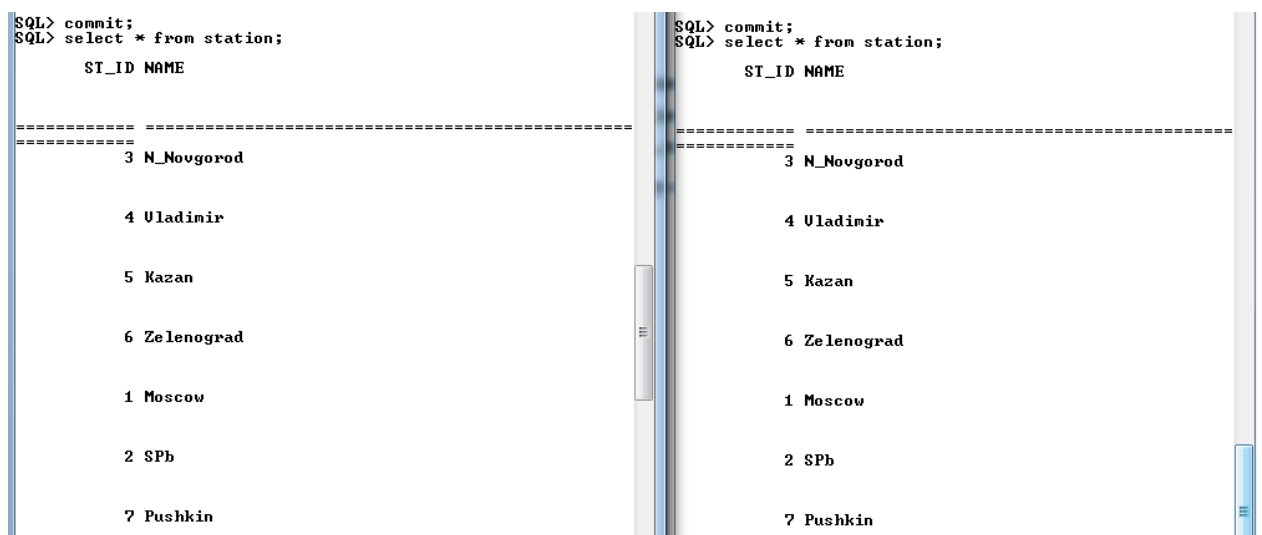


ST_ID	NAME
3	N_Novgorod
4	Uladimir
5	Kazan
6	Zelenograd
1	Moscow
7	Pushkin

ST_ID	NAME
3	N_Novgorod
4	Uladimir
5	Kazan
6	Zelenograd
2	SPb
7	Pushkin

Два клиента вставили по записи в таблицу – первый вставил запись о станции Москва, второй о станции СПб. Но до завершения транзакции клиенты не могут видеть изменений, которые совершил другой, видны лишь свои.

Завершим транзакции и проверим результат:



ST_ID	NAME
3	N_Novgorod
4	Uladimir
5	Kazan
6	Zelenograd
1	Moscow
2	SPb
7	Pushkin

ST_ID	NAME
3	N_Novgorod
4	Uladimir
5	Kazan
6	Zelenograd
1	Moscow
2	SPb
7	Pushkin

После завершения транзакций произведенные обоими клиентами изменения видны у всех.

- режим разрешения блокировок по wait

<pre>SQL> set transaction snapshot no wait; Commit current transaction <y/n>?y Committing. SQL> delete from station where st_id = 10; SQL> select * from station;</pre>	<pre>SQL> set transaction snapshot no wait; Commit current transaction <y/n>?y Committing. SQL> delete from station where st_id = 9; SQL> select * from station;</pre>																																								
<table><thead><tr><th>ST_ID</th><th>NAME</th></tr></thead><tbody><tr><td>8</td><td>Voronezh</td></tr><tr><td>3</td><td>N_Novgorod</td></tr><tr><td>4</td><td>Uladimir</td></tr><tr><td>5</td><td>Kazan</td></tr><tr><td>9</td><td>Volgograd</td></tr><tr><td>6</td><td>Zelenograd</td></tr><tr><td>1</td><td>Moscow</td></tr><tr><td>2</td><td>SPb</td></tr><tr><td>7</td><td>Pushkin</td></tr></tbody></table>	ST_ID	NAME	8	Voronezh	3	N_Novgorod	4	Uladimir	5	Kazan	9	Volgograd	6	Zelenograd	1	Moscow	2	SPb	7	Pushkin	<table><thead><tr><th>ST_ID</th><th>NAME</th></tr></thead><tbody><tr><td>8</td><td>Voronezh</td></tr><tr><td>3</td><td>N_Novgorod</td></tr><tr><td>4</td><td>Uladimir</td></tr><tr><td>5</td><td>Kazan</td></tr><tr><td>10</td><td>Iver</td></tr><tr><td>6</td><td>Zelenograd</td></tr><tr><td>1</td><td>Moscow</td></tr><tr><td>2</td><td>SPb</td></tr><tr><td>7</td><td>Pushkin</td></tr></tbody></table>	ST_ID	NAME	8	Voronezh	3	N_Novgorod	4	Uladimir	5	Kazan	10	Iver	6	Zelenograd	1	Moscow	2	SPb	7	Pushkin
ST_ID	NAME																																								
8	Voronezh																																								
3	N_Novgorod																																								
4	Uladimir																																								
5	Kazan																																								
9	Volgograd																																								
6	Zelenograd																																								
1	Moscow																																								
2	SPb																																								
7	Pushkin																																								
ST_ID	NAME																																								
8	Voronezh																																								
3	N_Novgorod																																								
4	Uladimir																																								
5	Kazan																																								
10	Iver																																								
6	Zelenograd																																								
1	Moscow																																								
2	SPb																																								
7	Pushkin																																								

При режиме no wait уровень SNAPSHOT позволяет производить изменения обоим клиентам.

Уровень SNAPSHOT TABLE STABILITY

Уровень SNAPSHOT TABLE STABILITY позволяет, как и SNAPSHOT, также видеть только те изменения, фиксация которых произошла не позднее момента старта этой транзакции. Но 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.

<pre>SQL> set transaction snapshot table stability; Commit current transaction <y/n>?y Committing. SQL> insert into station(st_id, name) values(8, 'Voronezh'); SQL></pre>	<pre>SQL> set transaction snapshot table stability; Commit current transaction <y/n>?y Committing. SQL> insert into station(st_id, name) values(9, 'Volgograd'); SQL></pre>
---	--

Первый клиент (слева) добавил запись о станции Воронеж, второй попытался добавить запись о станции Волгоград, однако это ему не удалось.

Теперь дадим первому клиенту завершить транзакцию:

```
SQL> set transaction snapshot table stability;
Commit current transaction <y/n>?y
Committing.
SQL> insert into station<st_id, name> values(8, 'Voronezh');
SQL> commit;
SQL> _
```

```
Commit current transaction <y/n>?y
Committing.
SQL> insert into station<st_id, name> values(9, 'Volograd');
SQL> select * from station;

  ST_ID NAME
-----
      3 N_Novgorod

      4 Vladimir

      5 Kazan

      9 Volograd

      6 Zelenograd

      1 Moscow

      2 SPb

      7 Pushkin
```

Теперь второму клиенту удалось добавить свою запись.

- режим разрешения блокировок no wait

```
SQL> set transaction snapshot table stability no wait;
SQL> insert into station<st_id, name> values(10, 'Iver');
SQL> _
```

```
SQL> set transaction snapshot table stability no wait;
Commit current transaction <y/n>?y
Committing.
SQL> insert into station<st_id, name> values(11, 'Dmitrov');
Statement failed, SQLSTATE = 40001
lock conflict on no wait transaction
-Acquire lock for relation <STATION> failed
SQL> _
```

В этом режиме при добавлении своей записи второму клиенту выдается сообщение об ошибке.

Вывод.

В ходе данной лабораторной работы был изучен механизм транзакций.

Также были изучены уровни изоляции транзакций в Firebird. Уровень изолированности транзакций — значение, определяющее уровень, при котором в транзакции допускаются несогласованные данные, то есть степень изолированности одной транзакции от другой. Более высокий уровень изолированности повышает точность данных, но при этом может снижаться количество параллельно выполняемых транзакций. С другой стороны, более низкий уровень изолированности позволяет выполнять больше параллельных транзакций, но снижает точность данных. Для разных пользователей можно назначать различные уровни, что позволяет управлять способностью видеть изменения в базе данных, т.е. задавать некий приоритет.