Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Программная инженерия
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Транзакции
тема

Преподаватель _____ А. Д. Вожжов
подпись, дата — инициалы, фамилия

Студент КИ23-17/1б, 032320521 _____ А. С. Лысаковский
номер группы, зачётной книжки — подпись, дата — инициалы, фамилия

Красноярск 2025

1 ВВЕДЕНИЕ

1.1 Цель работы

Изучить теоретический материал по теме «Транзакции». Выполнить задания.

1.2 Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

1 изучить теоретический материал по предложенной теме;

2 выполнить задание;

3 предоставить отчёт преподавателю.

1.3 Задание

Задание данной практической работы состоит из следующих частей:

4 Выполнить задания из главы 8 из книги на е-курсах.

2 ХОД РАБОТЫ

2.1 Задание 1

По умолчанию каждая SQL-команда, выполняемая в среде psql, образует отдельную транзакцию с уровнем изоляции Read Committed. Поэтому в тех экспериментах, когда одна из транзакций состоит только из единственной SQL-команды, можно не выполнять команды BEGIN и END. Конечно, если каждая из параллельных транзакций состоит из единственной SQL-команды, то хотя бы для одной из транзакций придется все же выполнить и команду BEGIN, иначе эксперимент не получится. В тексте главы были приведены примеры транзакций, в которых рассматривались команды SELECT ... FOR UPDATE и LOCK TABLE. Попробуйте повторить эти эксперименты с учетом описанного поведения PostgreSQL.

На рисунках с 1 по 6 показан результат выполнения задания.

```
demo=# BEGIN;
BEGIN
Время: 1,101 мс
demo=*# SELECT * FROM bookings WHERE book_ref = 'ABC123' FOR UPDATE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 ABC123   | 2016-10-13 21:00:00+07 |     42000.00
(1 строка)


Время: 0,581 мс
```

Рисунок 1 – Работа с транзакциями, часть 1

```
demo=# BEGIN;
BEGIN
Время: 1,101 мс
demo=*# SELECT * FROM bookings WHERE book_ref = 'ABC123' FOR UPDATE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 ABC123   | 2016-10-13 21:00:00+07 |     42000.00
(1 строка)


Время: 0,581 мс
```

Рисунок 2 – Работа с транзакциями, часть 2

```
demo=# UPDATE bookings SET total_amount = 1500.00 WHERE book_ref = 'ABC123';
UPDATE 1
```

Рисунок 3 – Работа с транзакциями, часть 3

```
demo=# BEGIN;
BEGIN
demo=*# LOCK TABLE bookings IN ACCESS EXCLUSIVE MODE;
LOCK TABLE
```

Рисунок 4 – Работа с транзакциями, часть 4

```
demo=# SELECT * FROM bookings;
```

Рисунок 5 – Работа с транзакциями, часть 5

```
demo=# SELECT * FROM bookings;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00000F   | 2016-09-02 06:12:00+07 |    265700.00
 000012   | 2016-09-11 12:02:00+07 |     37900.00
 000068   | 2016-10-13 17:27:00+07 |     18100.00
```

Рисунок 6 – Работа с транзакциями, часть 6

2.2 Задание 2

Транзакции, работающие на уровне изоляции Read Committed, видят только свои собственные обновления и обновления, зафиксированные параллельными транзакциями. При этом нужно учитывать, что иногда могут возникать ситуации, которые на первый взгляд кажутся парадоксальными, но на самом деле все происходит в строгом соответствии с этим принципом.

Воспользуемся таблицей «Самолеты» (aircrafts) или ее копией. Предположим, что мы решили удалить из таблицы те модели, дальность полета которых менее 2 000 км. В таблице представлена одна такая модель — Cessna 208 Caravan, имеющая дальность полета 1 200 км. Для выполнения удаления мы организовали транзакцию.

Однако параллельная транзакция, которая, причем, началась раньше, успела обновить таблицу таким образом, что дальность полета самолета Cessna 208 Caravan стала составлять 2 100 км, а вот для самолета Bombardier CRJ-200 она, напротив, уменьшилась до 1 900 км.

Таким образом, в результате выполнения операций обновления в таблице по-прежнему присутствует строка, удовлетворяющая первоначальному условию, т. е. значение атрибута range у которой меньше 2000.

Наша задача: проверить, будет ли в результате выполнения двух транзакций удалена какая-либо строка из таблицы.

Модифицируйте сценарий выполнения транзакций: в первой транзакции вместо фиксации изменений выполните их отмену с помощью команды ROLLBACK и посмотрите, будет ли удалена строка и какая конкретно.

На рисунках 7-12 показан результат выполнения задания.

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range < 2000;
 aircraft_code |       model        | range
---------------+--------------------+-------
 CN1           | Cessna 208 Caravan |  1200
(1 строка)


demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 2100
demo-*# WHERE aircraft_code = 'CN1';
UPDATE 1
demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 1900
demo-*# WHERE aircraft_code = 'CR2';
UPDATE 1
```

Рисунок 7 – Работа с транзакциями, часть 1

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range < 2000;
 aircraft_code |       model        | range
---------------+--------------------+-------
 CN1           | Cessna 208 Caravan |  1200
(1 строка)


demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 2100
demo-*# WHERE aircraft_code = 'CN1';
UPDATE 1
demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 1900
demo-*# WHERE aircraft_code = 'CR2';
UPDATE 1
```

Рисунок 8 – Работа с транзакциями, часть 2

```
demo=*# DELETE FROM aircrafts_tmp WHERE range < 2000;
DELETE 0
demo=*# END;
COMMIT
```

Рисунок 9 – Работа с транзакциями, часть 3

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range < 2000;
 aircraft_code |       model        | range
---------------+--------------------+-------
 CR2           | Bombardier CRJ-200 |  1900
(1 строка)


demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 2000
demo-*# WHERE aircraft_code = 'CN1';
UPDATE 1
demo=*#
demo=*# UPDATE aircrafts_tmp
demo-*# SEt range = 2100
demo-*# WHERE aircraft_code = 'CR2';
UPDATE 1
```

Рисунок 10 – Работа с транзакциями, часть 4

```
demo=# BEGIN
demo-# ;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range < 2000;
 aircraft_code |       model        | range
---------------+--------------------+-------
 CR2           | Bombardier CRJ-200 |  1900
(1 строка)


demo=*#
demo=*# DELETE FROM aircrafts_tmp WHERE range < 2000;
```

Рисунок 11 – Работа с транзакциями, часть 5

```
demo=*# DELETE FROM aircrafts_tmp WHERE range < 2000;
DELETE 1
```

Рисунок 12 – Работа с транзакциями, часть 6

2.3 Задание 3

Когда говорят о таком феномене, как потерянное обновление, то зачастую в качестве примера приводится операция UPDATE, в которой значение какого-то атрибута изменяется с применением арифметической операции. Например:

```
UPDATE aircrafts_tmp
SET range = range + 200
WHERE aircraft_code = 'CR2';
```

При выполнении двух и более подобных обновлений в рамках параллельных транзакций (например, на уровне изоляции Read Committed), все такие изменения будут учтены (как показано в тексте главы). Очевидно, что в этом случае потерянного обновления не происходит.

Предположим, что в одной транзакции будет просто присваиваться новое значение, например, так:

```
UPDATE aircrafts_tmp
SET range = 2100
WHERE aircraft_code = 'CR2';
```

А в параллельной транзакции будет выполняться аналогичная команда:

```
UPDATE aircrafts_tmp
SET range = 2500
WHERE aircraft_code = 'CR2';
```

Очевидно, что сохранится только одно из значений атрибута range. Можно ли говорить, что в такой ситуации имеет место потерянное обновление? Если оно имеет место, то что можно предпринять для его недопущения? Обоснуйте ваш ответ.

На рисунках с 13 по 16 показан результат выполнения задания.

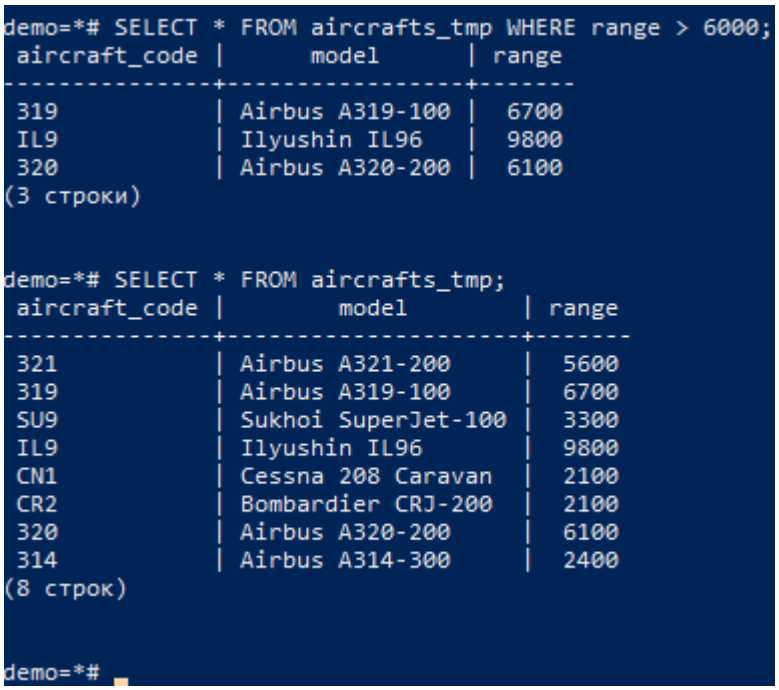

Рисунок 13 – Работа с транзакциями, часть 1

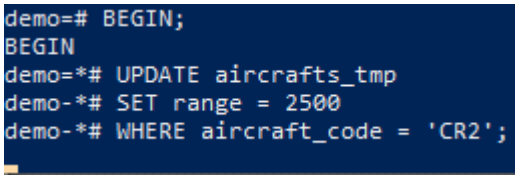

Рисунок 14 – Работа с транзакциями, часть 2

```
demo=*# UPDATE aircrafts_tmp
demo-*# SET range = 2500
demo-*# WHERE aircraft_code = 'CR2';
UPDATE 1
```

Рисунок 15 – Работа с транзакциями, часть 3

```
demo=# SELECT * FROM aircrafts_tmp;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 320           | Airbus A320-200      |  5800
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  2100
(7 строк)
```

Рисунок 16 – Работа с транзакциями, часть 4

2.4 Задание 4

На уровне изоляции транзакций Read Committed имеет место такой феномен, как чтение фантомных строк. Такие строки могут появляться в выборке как в результате добавления новых строк параллельной транзакцией, так и вследствие изменения ею значений атрибутов, участвующих в формировании условия выборки.

Покажем пример, иллюстрирующий вторую из указанных причин. Модифицируем его на работу с INSERT.

На рисунках с 17 по 22 показан результат выполнения задания.

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 2000 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  2100
 320           | Airbus A320-200      |  6100
 314           | Airbus A314-300      |  2400
(8 строк)
```

Рисунок 17 – Работа с транзакциями, часть 1

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 2000 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  2100
 320           | Airbus A320-200      |  6100
 314           | Airbus A314-300      |  2400
(8 строк)
```

Рисунок 18 – Работа с транзакциями, часть 2

```
demo=*# SELECT *
demo-*# FROM aircrafts_tmp
demo-*# WHERE range > 6000;
 aircraft_code |      model      | range
---------------+-----------------+-------
 319           | Airbus A319-100 |  6700
 IL9           | Ilyushin IL96   |  9800
 320           | Airbus A320-200 |  6100
(3 строки)
```

Рисунок 19 – Работа с транзакциями, часть 3

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 6000;
 aircraft_code |      model      | range
---------------+-----------------+-------
 319           | Airbus A319-100 |  6700
 IL9           | Ilyushin IL96   |  9800
 320           | Airbus A320-200 |  6100
(3 строки)
```

Рисунок 20 – Работа с транзакциями, часть 4

```
demo=# BEGIN;
BEGIN
demo=*# INSERT INTO aircrafts_tmp
demo-*# VALUES ('314', 'Airbus A314-300', 2400);
INSERT 0 1
demo=*# END;
COMMIT
```

Рисунок 21 – Работа с транзакциями, часть 5

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 6000;
 aircraft_code |      model      | range
---------------+-----------------+-------
 319           | Airbus A319-100 |  6700
 IL9           | Ilyushin IL96   |  9800
 320           | Airbus A320-200 |  6100
(3 строки)


demo=*# SELECT * FROM aircrafts_tmp;
 aircraft_code |        model        | range
---------------+---------------------+-------
 321           | Airbus A321-200     |  5600
 319           | Airbus A319-100     |  6700
 SU9           | Sukhoi SuperJet-100 |  3300
 IL9           | Ilyushin IL96       |  9800
 CN1           | Cessna 208 Caravan  |  2100
 CR2           | Bombardier CRJ-200  |  2100
 320           | Airbus A320-200     |  6100
 314           | Airbus A314-300     |  2400
(8 строк)


demo=*#
```

Рисунок 22 – Работа с транзакциями, часть 6

2.5 Задание 5

В тексте главы была рассмотрена команда SELECT ... FOR UPDATE, выполняющая блокировку на уровне отдельных строк. Организуйте две параллельные транзакции с уровнем изоляции Read Committed и выполните с ними ряд экспериментов. В первой транзакции заблокируйте некоторое множество строк, отбираемых с помощью условия WHERE. А во второй транзакции изменяйте условие выборки таким образом, чтобы выбираемое множество строк:

– являлось подмножеством множества строк, выбираемых в первой транзакции;

– являлось надмножеством множества строк, выбираемых в первой транзакции;

– пересекалось с множеством строк, выбираемых в первой транзакции;

– не пересекалось с множеством строк, выбираемых в первой транзакции.

Наблюдайте за поведением команд выборки в каждой транзакции. Попробуйте обобщить ваши наблюдения.

На рисунках с 23 по 30 показан результат выполнения задания.

```
demo=*# SELECT * FROM bookings WHERE book_ref = 'ABC123' FOR UPDATE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 ABC123   | 2016-10-13 21:00:00+07 |      1500.00
(1 строка)
```

Рисунок 23 – Работа с транзакциями, часть 1

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 3200 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 320           | Airbus A320-200      |  6100
(5 строк)
```

Рисунок 24 – Работа с транзакциями, часть 2

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 3300 FOR UPDATE;
```

Рисунок 25 – Работа с транзакциями, часть 3

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 3200 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 320           | Airbus A320-200      |  6100
(5 строк)
```

Рисунок 26 – Работа с транзакциями, часть 4

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 2000 FOR UPDATE;
```

Рисунок 27 – Работа с транзакциями, часть 5

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 3200 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 320           | Airbus A320-200      |  6100
(5 строк)
```

Рисунок 28 – Работа с транзакциями, часть 6

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range < 2500 FOR UPDATE;
 aircraft_code |       model          | range
---------------+----------------------+-------
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  2100
 314           | Airbus A314-300      |  2400
(3 строки)
```

Рисунок 29 – Работа с транзакциями, часть 7

```
demo=*# SELECT * FROM aircrafts_tmp WHERE range > 2000 FOR UPDATE;
 aircraft_code |        model         | range
---------------+----------------------+-------
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 SU9           | Sukhoi SuperJet-100  |  3300
 IL9           | Ilyushin IL96        |  9800
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  2100
 320           | Airbus A320-200      |  6100
 314           | Airbus A314-300      |  2400
(8 строк)
```

Рисунок 30 – Работа с транзакциями, часть 8

2.6 Задание 6

Самостоятельно ознакомьтесь с предложением FOR SHARE команды SELECT и выполните необходимые эксперименты. Используйте документацию: раздел 13.3.2 «Блокировки на уровне строк» и описание команды SELECT.

На рисунках с 30 по 38 показан результат выполнения задания.

```
demo=*# UPDATE bookings SET total_amount = 2000.00 WHERE book_ref = 'ABC123';
UPDATE 1
```

Рисунок 31 – Работа с транзакциями, часть 1

```
demo=# SELECT * FROM bookings WHERE total_amount > 500;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00000F   | 2016-09-02 06:12:00+07 |    265700.00
 000012   | 2016-09-11 12:02:00+07 |     37900.00
 000068   | 2016-10-13 17:27:00+07 |     18100.00
```

Рисунок 32 – Работа с транзакциями, часть 2

```
demo=*# SELECT * FROM bookings WHERE total_amount > 500 FOR SHARE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00000F   | 2016-09-02 06:12:00+07 |    265700.00
 000012   | 2016-09-11 12:02:00+07 |     37900.00
 000068   | 2016-10-13 17:27:00+07 |     18100.00
```

Рисунок 33 – Работа с транзакциями, часть 3

```
demo=# BEGIN;
BEGIN
demo=*# UPDATE bookings SET total_amount = 2000.00 WHERE book_ref = 'ABC123';
```

Рисунок 34 – Работа с транзакциями, часть 4

```
demo=*# UPDATE bookings SET total_amount = 2000.00 WHERE book_ref = 'ABC123';
UPDATE 1
```

Рисунок 35 – Работа с транзакциями, часть 5

```
demo=*# SELECT * FROM bookings WHERE total_amount > 500 FOR SHARE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00000F   | 2016-09-02 06:12:00+07 |    265700.00
 000012   | 2016-09-11 12:02:00+07 |     37900.00
 000068   | 2016-10-13 17:27:00+07 |     18100.00
```

Рисунок 36 – Работа с транзакциями, часть 6

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM bookings WHERE book_ref = 'ABC123' FOR UPDATE;
```

Рисунок 37 – Работа с транзакциями, часть 7

```
demo=*# SELECT * FROM bookings WHERE book_ref = 'ABC123' FOR UPDATE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 ABC123   | 2016-10-13 21:00:00+07 |      1500.00
(1 строка)
```

Рисунок 38 – Работа с транзакциями, часть 8

2.7 Задание 7

В тексте главы для иллюстрации изучаемых концепций мы создавали только две параллельные транзакции. Попробуйте воспроизвести

представленные эксперименты, создав три или даже четыре параллельные транзакции.

На рисунках с 39 по 48 показан результат выполнения задания.

```
demo=*# BEGIN;SELECT * FROM bookings WHERE total_amount < 15000 FOR UPDATE;
ПРЕДУПРЕЖДЕНИЕ:  транзакция уже выполняется
BEGIN
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00044D   | 2016-09-27 03:24:00+07 |      6000.00
 0004B0   | 2016-09-25 12:00:00+07 |     12000.00
 00053F   | 2016-10-04 06:15:00+07 |      6000.00
```

Рисунок 39 – Работа с транзакциями, часть 1

```
demo=# BEGIN;
BEGIN
demo=*#
demo=*# SELECT * FROM bookings WHERE total_amount > 1000 FOR UPDATE;
```

Рисунок 40 – Работа с транзакциями, часть 2

```
demo=*# UPDATE bookings SET total_amount = 1500.00 WHERE book_ref = '00044D';
```

Рисунок 41 – Работа с транзакциями, часть 3

```
demo=*# SELECT * FROM bookings WHERE total_amount > 1000 FOR UPDATE;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 00000F   | 2016-09-02 06:12:00+07 |    265700.00
 000012   | 2016-09-11 12:02:00+07 |     37900.00
 000068   | 2016-10-13 17:27:00+07 |     18100.00
```

Рисунок 42 – Работа с транзакциями, часть 4

```
demo=*# UPDATE bookings SET total_amount = 1500.00 WHERE book_ref = '00044D';
UPDATE 1
```

Рисунок 43 – Работа с транзакциями, часть 5

```
demo=# BEGIN;
BEGIN
demo=*# BEGIN;
ПРЕДУПРЕЖДЕНИЕ:  транзакция уже выполняется
BEGIN
demo=*# LOCK TABLE bookings IN ACCESS EXCLUSIVE MODE;
LOCK TABLE
```

Рисунок 44 – Работа с транзакциями, часть 6

```
demo=# BEGIN;
BEGIN
demo=*# SELECT * FROM bookings WHERE total_amount > 500;
```

Рисунок 45 – Работа с транзакциями, часть 7

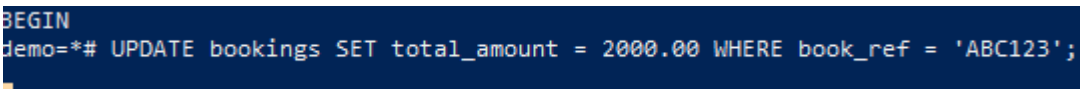

Рисунок 46 – Работа с транзакциями, часть 8

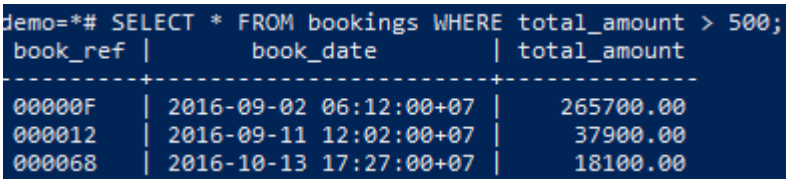

Рисунок 47 – Работа с транзакциями, часть 9

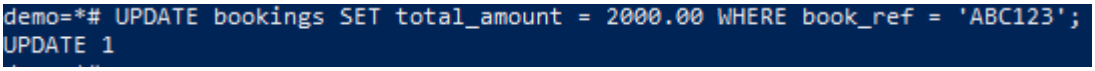

Рисунок 48 – Работа с транзакциями, часть 10

2.8 Задание 8

Задание. В тексте главы была рассмотрена транзакция для выполнения бронирования билетов. Для нее был выбран уровень изоляции Read Committed. Как вы думаете, если одновременно будут производиться несколько операций бронирования, то, может быть, имеет смысл «ужесточить» уровень изоляции до Serializable? Или нет необходимости это делать? Обдумайте и вариант с использованием явных блокировок. Обоснуйте ваш ответ.

Ответ. Ужесточение уровня изоляции до «Serializable» имеет смысл, если система бронирования подвержена высокой конкуренции и критически важно избежать любых аномалий. Использование явных блокировок (например, SELECT FOR UPDATE) — это более гибкий подход, который позволяет сохранить высокую производительность при «Read Committed». Предпочтительнее использовать явные блокировки.

2.9 Задание 9

В разделе документации 13.2.3 «Уровень изоляции Serializable» сказано, что если поиск в таблице осуществляется последовательно, без использования индекса, тогда на всю таблицу накладывается так называемая предикатная блокировка. Такой подход приводит к увеличению числа сбоев сериализации. В качестве контрмеры можно попытаться использовать индексы. Конечно, если таблица совсем небольшая, то может и не получиться заставить PostgreSQL использовать поиск по индексу. Тем не менее давайте выполним следующий эксперимент.

Повторив эксперимент, необходимо ответить на вопросы. Как вы думаете, почему это удалось? Обосновывая ваш ответ, примите во внимание тот результат, который был бы получен при последовательном выполнении транзакций.

На рисунках с 49 по 56 показан результат выполнения задания.

```
demo=# CREATE TABLE modes AS
demo-# SELECT num::integer, 'LOW' || num::text AS mode
demo-# FROM generate_series( 1, 100000 ) AS gen_ser( num )
demo-# UNION ALL
demo-# SELECT num::integer, 'HIGH' || ( num - 100000 )::text AS mode
demo-# FROM generate_series( 100001, 200000 ) AS gen_ser( num );
SELECT 200000
```

Рисунок 49 – Работа с транзакциями, часть 1

```
demo=# SELECT *
demo-# FROM modes
demo-# WHERE mode IN ( 'LOW1', 'HIGH1' );
  num   | mode
--------+-------
      1 | LOW1
 100001 | HIGH1
(2 строки)
```

Рисунок 50 – Работа с транзакциями, часть 2

```
demo=# SELECT *
demo-# FROM modes
demo-# WHERE mode IN ( 'LOW1', 'HIGH1' );
  num   | mode
--------+-------
      1 | LOW1
 100001 | HIGH1
(2 строки)
```

Рисунок 51 – Работа с транзакциями, часть 3

```
demo=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
demo=*# UPDATE modes
demo-*# SET mode = 'HIGH1'
demo-*# WHERE num = 1;
UPDATE 1
```

Рисунок 52 – Работа с транзакциями, часть 4

```
demo=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
demo=*# UPDATE modes
demo-*# SET mode = 'LOW1'
demo-*# WHERE num = 100001;
UPDATE 1
```

Рисунок 53 – Работа с транзакциями, часть 5

```
demo=*# COMMIT;
COMMIT
```

Рисунок 54 – Работа с транзакциями, часть 6

```
demo=*# COMMIT;
COMMIT
```

Рисунок 55 – Работа с транзакциями, часть 7

2.10 Задание 10

В тексте главы был рассмотрен пример транзакции над таблицами базы данных «Авиаперевозки». Давайте теперь создадим две параллельные транзакции и выполним их с уровнем изоляции Serializable. Отправим также двоих пассажиров теми же самыми рейсами, что и ранее, но операции распределим между двумя транзакциями. Отличие заключается в том, что в начале транзакции будут выполняться выборки из таблицы ticket_flights. Для упрощения ситуации не будем предварительно проверять наличие свободных мест, т.к. сейчас для нас важно не это.

Задание 1. Попытайтесь объяснить, почему транзакции не удалось сериализовать. Что можно сделать, чтобы удалось зафиксировать обе транзакции? Одно из возможных решений — понизить уровень изоляции. Другим решением может быть создание индекса по столбцу flight_id для таблицы ticket_flights. Почему создание индекса может помочь? Обратитесь за разъяснениями к разделу документации 13.2.3 «Уровень изоляции Serializable».

Задание 2. В первой транзакции условие в команде SELECT такое: ... WHERE flight_id = 13881. В команде вставки в таблицу ticket_flights значение поля flight_id также равно 13881. Во второй транзакции в этих же командах используется значение 5572. Поменяйте местами значения в командах SELECT и повторите эксперименты, выполнив транзакции параллельно с уровнем изоляции Serializable. Почему сейчас наличие индекса не помогает зафиксировать обе транзакции? Вспомните, что аномалия сериализации — это ситуация, когда параллельное выполнение транзакций приводит к результату, невозможному ни при каком из вариантов упорядочения этих же транзакций при их последовательном выполнении.

На рисунках с 57 по 60 показан результат работы.

```
demo=*# INSERT INTO bookings ( book_ref, book_date, total_amount )
demo-*# VALUES ( 'ABC123', bookings.now(), 0 );
INSERT 0 1
demo=*# INSERT INTO tickets
demo-*# ( ticket_no, book_ref, passenger_id, passenger_name )
demo-*# VALUES ( '9991234567890', 'ABC123', '1234 123456', 'IVAN PETROV' );
INSERT 0 1
demo=*# INSERT INTO ticket_flights
demo-*# ( ticket_no, flight_id, fare_conditions, amount )
demo-*# VALUES ( '9991234567890', 13881, 'Business', 12500 );
INSERT 0 1
demo=*# UPDATE bookings
demo-*# SET total_amount = 12500
demo-*# WHERE book_ref = 'ABC123';
UPDATE 1
demo=*# COMMIT;
COMMIT
```

Рисунок 56 – Работа с транзакциями, часть 1

```
demo=*# INSERT INTO bookings ( book_ref, book_date, total_amount )
demo-*# VALUES ( 'ABC123', bookings.now(), 0 );
INSERT 0 1
demo=*# INSERT INTO tickets
demo-*# ( ticket_no, book_ref, passenger_id, passenger_name )
demo-*# VALUES ( '9991234567890', 'ABC123', '1234 123456', 'IVAN PETROV' );
INSERT 0 1
demo=*# INSERT INTO ticket_flights
demo-*# ( ticket_no, flight_id, fare_conditions, amount )
demo-*# VALUES ( '9991234567890', 13881, 'Business', 12500 );
INSERT 0 1
demo=*# UPDATE bookings
demo-*# SET total_amount = 12500
demo-*# WHERE book_ref = 'ABC123';
UPDATE 1
demo=*# COMMIT;
COMMIT
```

Рисунок 57 – Работа с транзакциями, часть 2

```
demo=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
demo=*# SELECT *
demo-*# FROM ticket_flights
demo-*# WHERE flight_id = 5572;
   ticket_no   | flight_id | fare_conditions |  amount
---------------+-----------+-----------------+----------
 0005433847924 |      5572 | Business        | 99800.00
 0005433847922 |      5572 | Business        | 99800.00
 0005435282817 |      5572 | Business        | 99800.00
```

Рисунок 58 – Работа с транзакциями, часть 3

```
demo=*# INSERT INTO bookings ( book_ref, book_date, total_amount )
demo-*# VALUES ( 'ABC456', bookings.now(), 0 );
INSERT 0 1
demo=*# INSERT INTO tickets
demo-*# ( ticket_no, book_ref, passenger_id, passenger_name )
demo-*# VALUES ( '9991234567891', 'ABC456', '4321 654321', 'PETR IVANOV' );
INSERT 0 1
demo=*# INSERT INTO ticket_flights
demo-*# ( ticket_no, flight_id, fare_conditions, amount )
demo-*# VALUES ( '9991234567891', 5572, 'Business', 12500 );
INSERT 0 1
demo=*# UPDATE bookings
demo-*# SET total_amount = 12500
demo-*# WHERE book_ref = 'ABC456';
UPDATE 1
demo=*# COMMIT;
COMMIT
```

Рисунок 59 – Работа с транзакциями, часть 4

```
demo=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
demo=*# SELECT * FROM ticket_flights WHERE flight_id = 5572 LIMIT 10;
   ticket_no   | flight_id | fare_conditions |  amount
---------------+-----------+-----------------+----------
 0005433847914 |      5572 | Economy         | 33300.00
 0005432081640 |      5572 | Economy         | 33300.00
 0005433847911 |      5572 | Economy         | 33300.00
 0005433847898 |      5572 | Economy         | 33300.00
 0005433847920 |      5572 | Economy         | 33300.00
 0005433847878 |      5572 | Economy         | 33300.00
 0005435282626 |      5572 | Economy         | 33300.00
 0005433847921 |      5572 | Economy         | 33300.00
 0005433847905 |      5572 | Economy         | 33300.00
 0005435282781 |      5572 | Economy         | 33300.00
(10 строк)


demo=*# INSERT INTO bookings (book_ref, book_date, total_amount)
demo-*# VALUES ('ABC123', bookings.now(), 0);
INSERT 0 1
demo=*# INSERT INTO tickets
demo-*# (ticket_no, book_ref, passenger_id, passenger_name)
demo-*# VALUES ('9991234567890', 'ABC123', '1234 123456', 'IVAN PETROV');
INSERT 0 1
demo=*# INSERT INTO ticket_flights
demo-*# (ticket_no, flight_id, fare_conditions, amount)
demo-*# VALUES ('9991234567890', 13881, 'Business', 12500);
INSERT 0 1
demo=*# UPDATE bookings
demo-*# SET total_amount = 12500
demo-*# WHERE book_ref = 'ABC123';
UPDATE 1
```

Рисунок 60 – Работа с транзакциями, часть 5

```
demo=*# COMMIT;
COMMIT
```

Рисунок 61 – Работа с транзакциями, часть 6

```
demo=*# COMMIT;
COMMIT
```

Рисунок 62 – Работа с транзакциями, часть 7

```
demo=*# COMMIT;
ОШИБКА:  не удалось сериализовать доступ из-за зависимостей чтения/записи между транзакциями
ПОДРОБНОСТИ:  Reason code: Canceled on identification as a pivot, during commit attempt.
ПОДСКАЗКА:  Транзакция может завершиться успешно при следующей попытке.
```

Рисунок 63 – Работа с транзакциями, часть 8

3 ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Транзакции». Все поставленные цели и задачи были выполнены.