|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | | | | | |

# ВВЕДЕНИЕ

## Цель работы

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

## Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

1. изучить теоретический материал по предложенной теме;
2. выполнить задание;
3. предоставить отчёт преподавателю.

## Задание

Задание данной практической работы состоит из следующих частей:

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

# ХОД РАБОТЫ

## 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.

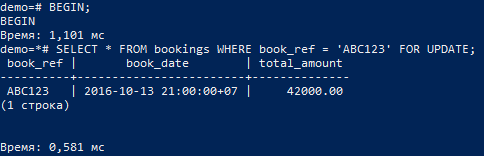


Рисунок 1 – Работа с транзакциями, часть 1

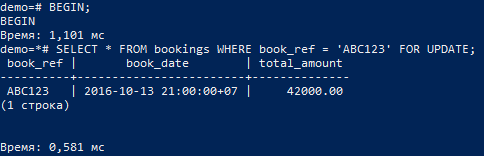


Рисунок 2 – Работа с транзакциями, часть 2



Рисунок 3 – Работа с транзакциями, часть 3



Рисунок 4 – Работа с транзакциями, часть 4



Рисунок 5 – Работа с транзакциями, часть 5

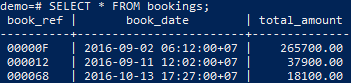


Рисунок 6 – Работа с транзакциями, часть 6

## 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.

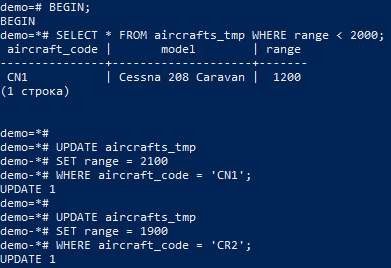


Рисунок 7 – Работа с транзакциями, часть 1

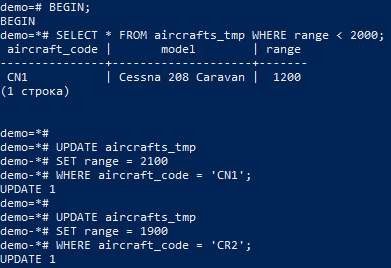


Рисунок 8 – Работа с транзакциями, часть 2



Рисунок 9 – Работа с транзакциями, часть 3

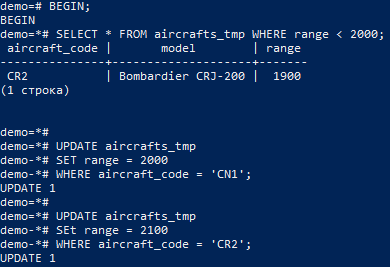


Рисунок 10 – Работа с транзакциями, часть 4

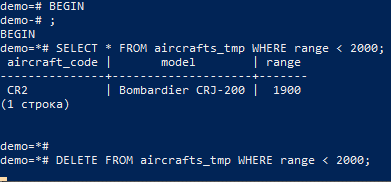


Рисунок 11 – Работа с транзакциями, часть 5



Рисунок 12 – Работа с транзакциями, часть 6

## 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.

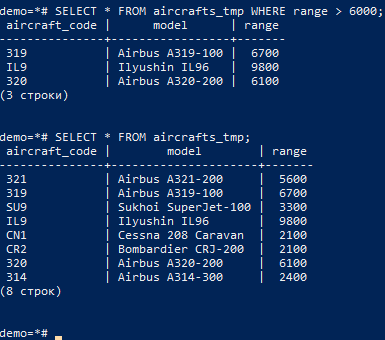


Рисунок 13 – Работа с транзакциями, часть 1

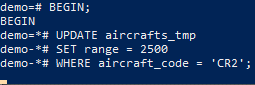


Рисунок 14 – Работа с транзакциями, часть 2



Рисунок 15 – Работа с транзакциями, часть 3

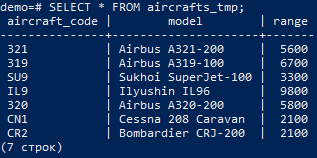


Рисунок 16 – Работа с транзакциями, часть 4

## 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.

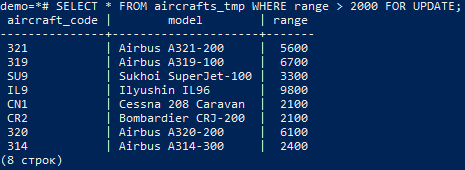


Рисунок 17 – Работа с транзакциями, часть 1

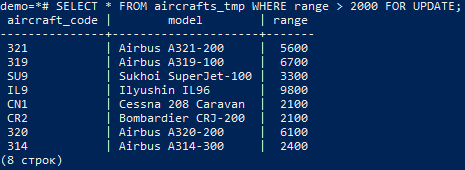


Рисунок 18 – Работа с транзакциями, часть 2

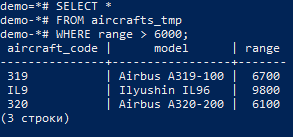


Рисунок 19 – Работа с транзакциями, часть 3

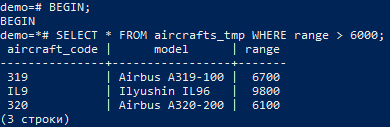


Рисунок 20 – Работа с транзакциями, часть 4

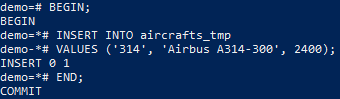


Рисунок 21 – Работа с транзакциями, часть 5

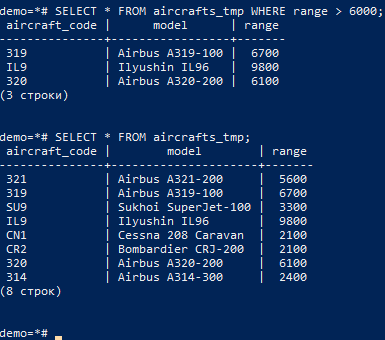


Рисунок 22 – Работа с транзакциями, часть 6

## 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.

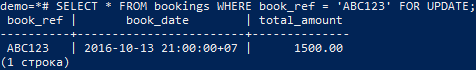


Рисунок 23 – Работа с транзакциями, часть 1

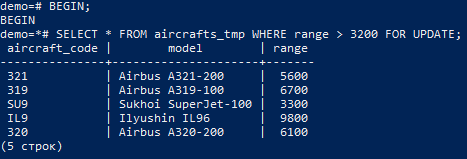


Рисунок 24 – Работа с транзакциями, часть 2



Рисунок 25 – Работа с транзакциями, часть 3

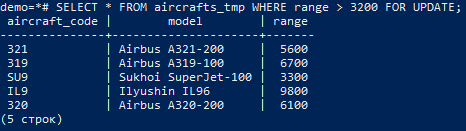


Рисунок 26 – Работа с транзакциями, часть 4



Рисунок 27 – Работа с транзакциями, часть 5

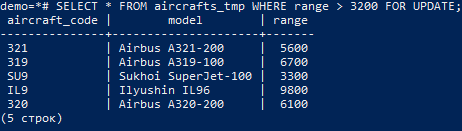


Рисунок 28 – Работа с транзакциями, часть 6

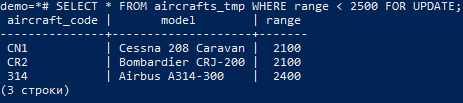


Рисунок 29 – Работа с транзакциями, часть 7

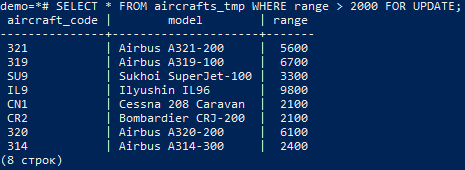


Рисунок 30 – Работа с транзакциями, часть 8

## 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.



Рисунок 31 – Работа с транзакциями, часть 1

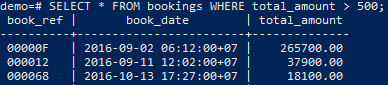


Рисунок 32 – Работа с транзакциями, часть 2

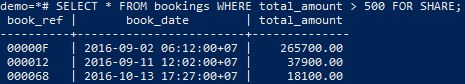


Рисунок 33 – Работа с транзакциями, часть 3



Рисунок 34 – Работа с транзакциями, часть 4



Рисунок 35 – Работа с транзакциями, часть 5

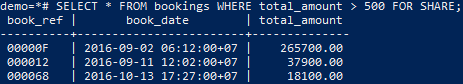


Рисунок 36 – Работа с транзакциями, часть 6



Рисунок 37 – Работа с транзакциями, часть 7

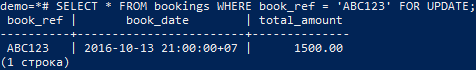


Рисунок 38 – Работа с транзакциями, часть 8

## 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.

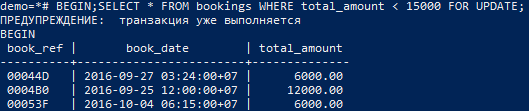


Рисунок 39 – Работа с транзакциями, часть 1

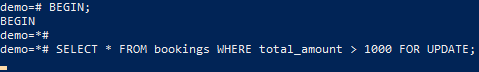


Рисунок 40 – Работа с транзакциями, часть 2



Рисунок 41 – Работа с транзакциями, часть 3

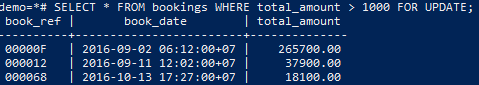


Рисунок 42 – Работа с транзакциями, часть 4



Рисунок 43 – Работа с транзакциями, часть 5

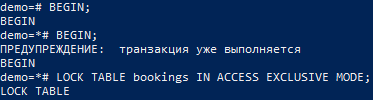


Рисунок 44 – Работа с транзакциями, часть 6



Рисунок 45 – Работа с транзакциями, часть 7



Рисунок 46 – Работа с транзакциями, часть 8

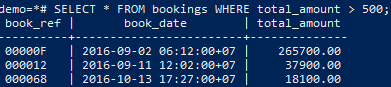


Рисунок 47 – Работа с транзакциями, часть 9



Рисунок 48 – Работа с транзакциями, часть 10

## 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.

## 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.

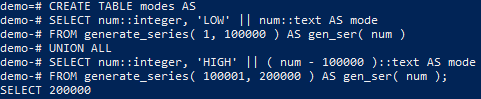


Рисунок 49 – Работа с транзакциями, часть 1

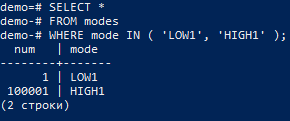


Рисунок 50 – Работа с транзакциями, часть 2

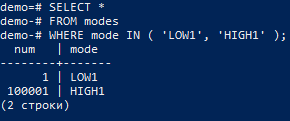


Рисунок 51 – Работа с транзакциями, часть 3

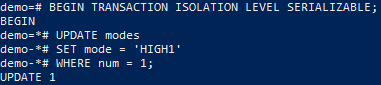


Рисунок 52 – Работа с транзакциями, часть 4

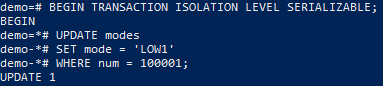


Рисунок 53 – Работа с транзакциями, часть 5



Рисунок 54 – Работа с транзакциями, часть 6



Рисунок 55 – Работа с транзакциями, часть 7

## 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.

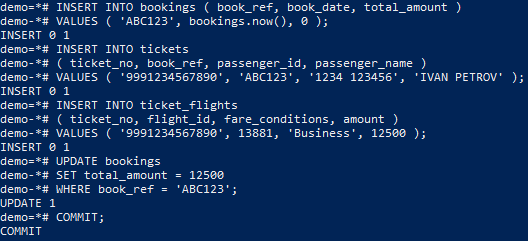


Рисунок 56 – Работа с транзакциями, часть 1

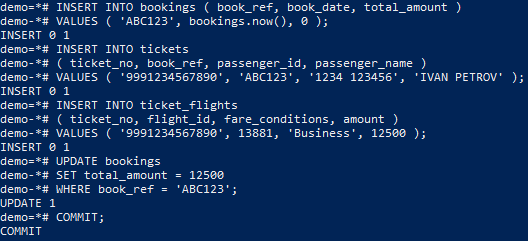


Рисунок 57 – Работа с транзакциями, часть 2

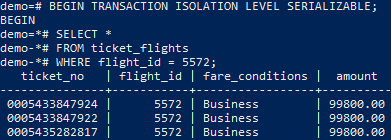


Рисунок 58 – Работа с транзакциями, часть 3

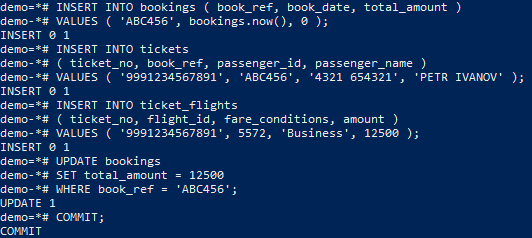


Рисунок 59 – Работа с транзакциями, часть 4

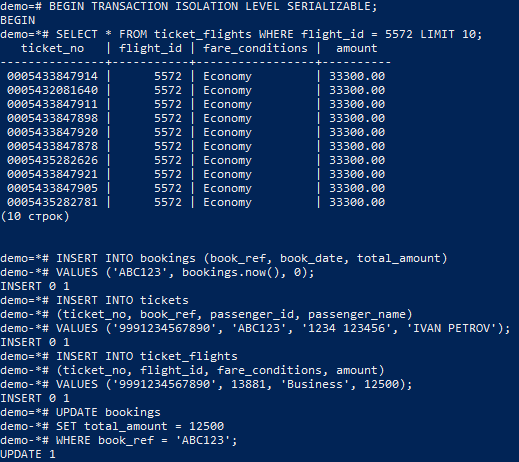


Рисунок 60 – Работа с транзакциями, часть 5



Рисунок 61 – Работа с транзакциями, часть 6



Рисунок 62 – Работа с транзакциями, часть 7



Рисунок 63 – Работа с транзакциями, часть 8

# ЗАКЛЮЧЕНИЕ

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