|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий (ИТ)

Кафедра Инструментального и прикладного программного обеспечения

(ИиППО)

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №4**

**по дисциплине**

«Технологии обработки транзакций клиент-серверных приложений»

Выполнил студент группы ИКБО-20-21 Сидоров С. Д.

Принял Маличенко С.В.

Москва 2024

**ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ**

**Уровень Serializable:**

1. В первом сеансе начните новую транзакцию с уровнем изоляции Serializable. Вычислите количество заказов с суммой 20 000 рублей. Добавьте новый заказ на 30 000 рублей. Получите снимок транзакции.
2. Во втором сеансе начните новую транзакцию с уровнем изоляции Serializable. Импортируйте снимок из первой транзакции. Вычислите количество заказов с суммой 30 000 рублей.
3. В первом сеансе вычислите количество заказов с суммой 20 000 рублей. Вычислите количество заказов с суммой 30 000 рублей.
4. Во втором сеансе добавьте новый заказ на 20 000 рублей и снова вычислите количество заказов с суммой 30 000 рублей. Вычислите количество заказов с суммой 20 000 рублей.
5. Зафиксируйте транзакции в обоих сеансах. Соответствует ли результат ожиданиями? Что вам дал импорт снимка из первой транзакции?

**Уровень Repeatable Read:**

1. В первом сеансе начните новую транзакцию с уровнем изоляции Repeatable Read. Вычислите количество заказов с суммой 20 000 рублей. Добавьте новый заказ на 30 000 рублей. Получите снимок транзакции.
2. Во втором сеансе начните новую транзакцию с уровнем изоляции Repeatable Read. Импортируйте снимок из первой транзакции. Вычислите количество заказов с суммой 30 000 рублей.
3. В первом сеансе вычислите количество заказов с суммой 20 000 рублей. Вычислите количество заказов с суммой 30 000 рублей.
4. Во втором сеансе добавьте новый заказ на 20 000 рублей и снова вычислите количество заказов с суммой 30 000 рублей. Вычислите количество заказов с суммой 20 000 рублей.
5. Зафиксируйте транзакции в обоих сеансах. Соответствует ли результат ожиданиями? Что вам дал импорт снимка из первой транзакции?

**РЕФЕРАТ**

Отчёт 14 страниц, 11 рисунков, 5 источников, 1 приложение.

POSTGRESQL, УРОВЕНЬ ИЗОЛЯЦИИ, ТРАНЗАКЦИИ, СНИМОК ДАННЫХ, SQL

Объектом разработки является база данных.

Цель работы – работа с экспортом снимков данных.

В процессе работы проводилось практическое использование транзакций с экспортом снимков данных.

Результатом являются сведения по работе транзакций с экспортом снимков данных.

**ВВЕДЕНИЕ**

[ВВЕДЕНИЕ 6](#_Toc159964595)

[1. ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ 7](#_Toc159964596)

[1.1 Уровень Serializable 7](#_Toc159964597)

[1.2 Уровень Repeatable Read 8](#_Toc159964598)

[ЗАКЛЮЧЕНИЕ 9](#_Toc159964599)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 10](#_Toc159964600)

[ПРИЛОЖЕНИЕ А 11](#_Toc159964601)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

БД – база данных;

СУБД – система управления базами данных.

**ВВЕДЕНИЕ**

В мире баз данных PostgreSQL, снимки данных играют ключевую роль в обеспечении надежности и восстановления информации. Способность сохранять консистентные копии базы данных на определенный момент времени обеспечивает защиту от потери данных, ошибок и несанкционированных изменений. Снимки предоставляют эффективный механизм для создания точных копий данных, сохраняя их в целостности. В данном контексте их важность раскрывается как инструмент обеспечения бизнес-непрерывности и безопасности данных, подчеркивая решающую роль снимков данных в стратегиях управления информационными ресурсами.

Целью данной работы является практическое исследование аспектов работы с экспортированием и импортированием снимков данных в СУБД PostgreSQL.

1. **ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ**
   1. **Уровень Serializable**

Перед началом работы база данных была приведена к изначальному состоянию. В первой сессии была начата транзакция с уровнем изоляции Serializable. Внутри данной транзакции было вычислено число заказов с суммой равной 20000, которых в базе не оказалось. После чего был создан заказ с суммой равной 30000 и был получен идентификатор снимка, который будет передан в сессию 2 в дальнейшем. Результат этих действий представлен на рисунке 1.

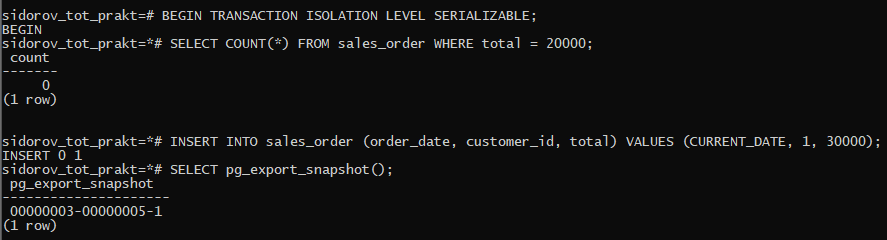


Рисунок 1 – Начало транзакции, добавление заказа и экспорт снимка данных в сессии 1

Далее была открыта сессия 2, в которой также была начата транзакция с уровнем изоляции Serializable. В данную транзакцию был экспортирован снимок данных из первой сессии, а также было вычислено количество заказов с суммой равной 30000, что отображено на рисунке 2.

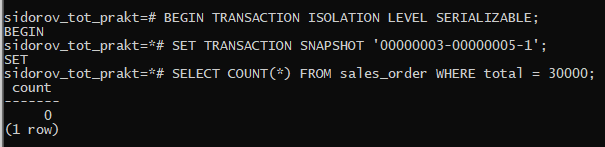


Рисунок 2 – Начало транзакции, импорт снимка данных и вычисление заказов в сессии 2

После этого была произведена проверка наличия вставленных данных в сессии 1, которая показала наличие записи о заказе на сумму 30000, что показано на рисунке 3.

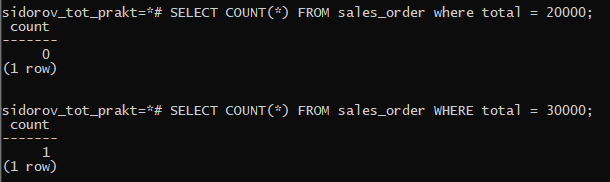


Рисунок 3 – Получение данных об определенных заказах в сессии 1

После в сессии 2 было произведено добавление заказа на сумму 20000, а также была произведена проверка на наличие заказов с суммой 30000 и 20000, которая показала отсутствие первых и наличие вторых, что представлено на рисунке 4.

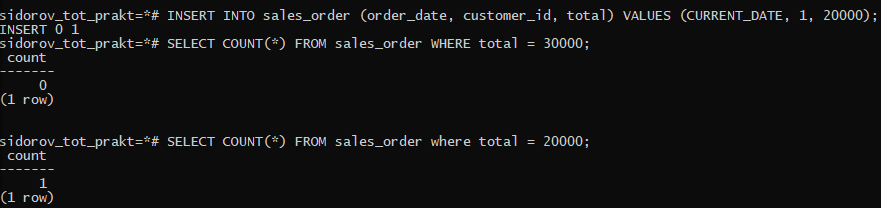


Рисунок 4 – Обновление данных о заказах и получение данных о заказах в сессии 2

В дальнейшем была успешно произведена фиксация изменений первой транзакции, что видно на рисунке 5, а также фиксация транзакции в сессии 2, которая выдала ошибку так, как данные транзакции невозможно сериализовать, а экспорт снимка данных позволяет видеть состояние системы до начала транзакции, а не данные, измененные в ней.



Рисунок 5 – Результат фиксации изменений сессии 1

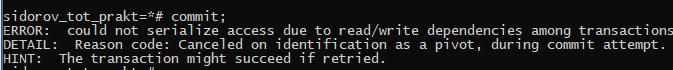


Рисунок 6 – Результат попытки фиксации изменений сессии 2

* 1. **Уровень Repeatable Read**

Перед началом работы база данных была приведена к изначальному состоянию. В первой сессии была начата транзакция с уровнем изоляции Repeatable Read. Внутри данной транзакции было вычислено число заказов с суммой равной 20000, которых в базе не оказалось. После чего был создан заказ с суммой равной 30000 и был получен идентификатор снимка, который будет передан в сессию 2 в дальнейшем. Результат этих действий представлен на рисунке 7.

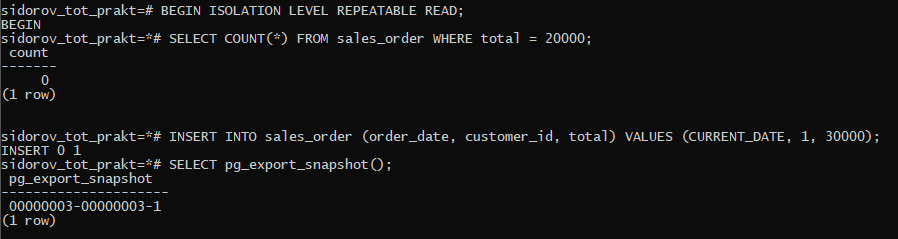


Рисунок 7 – Начало транзакции, добавление заказа и экспорт снимка данных в сессии 1

Далее была открыта сессия 2, в которой также была начата транзакция с уровнем изоляции Repeatable Read. В данную транзакцию был экспортирован снимок данных из первой сессии, а также было вычислено количество заказов с суммой равной 30000, что отображено на рисунке 2.

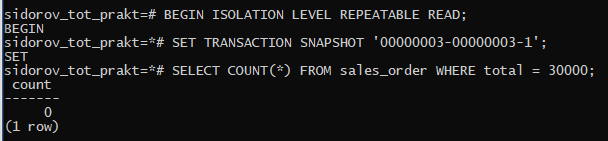


Рисунок 8 – Начало транзакции, импорт снимка данных и вычисление заказов в сессии 2

После этого была произведена проверка наличия вставленных данных в сессии 1, которая показала наличие записи о заказе на сумму 30000, что показано на рисунке 8.

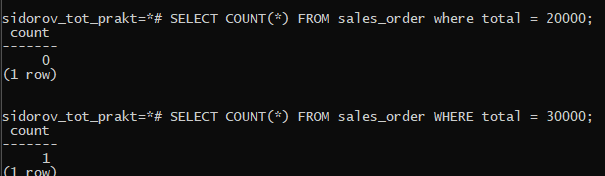


Рисунок 8 – Получение данных об определенных заказах в сессии 1

После в сессии 2 было произведено добавление заказа на сумму 20000, а также была произведена проверка на наличие заказов с суммой 30000 и 20000, которая показала отсутствие первых и наличие вторых, что представлено на рисунке 9.

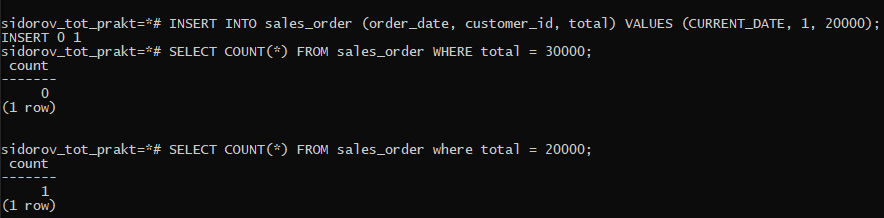


Рисунок 9 – Обновление данных о заказах и получение данных о заказах в сессии 2

В дальнейшем были успешно зафиксированы обе транзакции, что представлено на рисунках 10, 11. Как и ожидалось при экспорте снимка данных в данном сценарии работы изменений не произошло, так как экспорт снимка данных позволяет перенести только изначальное состояние системы, а не изменения внутри транзакции.

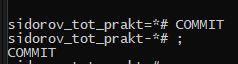


Рисунок 10 – Результат фиксации изменений сессии 1



Рисунок 11 – Результат фиксации изменений сессии 2

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы было осуществлено детальное изучение процесса работы со снимками данных в системе управления базами данных PostgreSQL. Была проведена работа с использованием экспорта снимка данных в транзакциях с уровнем изоляции Serializable и Repeatable Read, были получены знания об особенностях работы со снимками данных.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. PostgreSQL: Официальная документация [Электронный ресурс] – URL: https://www.postgresql.org/docs/ (дата обращения: 14.02.2024).
2. PostgreSQL Tutorial for Beginners [Электронный ресурс] – URL: https://www.tutorialspoint.com/postgresql/index.htm (дата обращения: 14.02.2024).
3. PostgreSQL: Википедия [Электронный ресурс] – URL: https://en.wikipedia.org/wiki/PostgreSQL (дата обращения: 14.02.2024).
4. Лузанов П.В. и др. Postgres. Первое знакомство. [Электронный ресурс] – URL: https://postgrespro.ru/education/books/introbook (дата обращения: 22.02.2024)
5. Новиков Б. А. Лекции Основы технологий баз данных. [Электронный ресурс] – URL: https://postgrespro.ru/education/university/dbtech (дата обращения: 22.02.2024)

# ПРИЛОЖЕНИЕ А

**ФРАГМЕНТЫ КОДА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ**

Листинг 1 - SQL код для выполнения задания 1 практической работы

--Сессия 1--

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SELECT COUNT(\*) FROM sales\_order WHERE total = 20000;

INSERT INTO sales\_order (order\_date, customer\_id, total) VALUES (CURRENT\_DATE, 1, 30000);

SELECT pg\_export\_snapshot();

--Сессия 2--

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;

SET TRANSACTION SNAPSHOT '00000003-00000003-1';

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

--Сессия 1--

SELECT COUNT(\*) FROM sales\_order where total = 20000;

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

--Сессия 2--

INSERT INTO sales\_order (order\_date, customer\_id, total) VALUES (CURRENT\_DATE, 1, 20000);

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

SELECT COUNT(\*) FROM sales\_order where total = 20000;

--Сессия 1--

COMMIT;

--Сессия 2--

COMMIT;

Листинг 2 – SQL код для выполнения задания 2 практической работы

--Сессия 1--

BEGIN ISOLATION LEVEL REPEATABLE READ;

SELECT COUNT(\*) FROM sales\_order WHERE total = 20000;

INSERT INTO sales\_order (order\_date, customer\_id, total) VALUES (CURRENT\_DATE, 1, 30000);

SELECT pg\_export\_snapshot();

--Сессия 2--

BEGIN ISOLATION LEVEL REPEATABLE READ;

SET TRANSACTION SNAPSHOT '00000003-00000003-1';

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

--Сессия 1--

SELECT COUNT(\*) FROM sales\_order where total = 20000;

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

--Сессия 2--

INSERT INTO sales\_order (order\_date, customer\_id, total) VALUES (CURRENT\_DATE, 1, 20000);

SELECT COUNT(\*) FROM sales\_order WHERE total = 30000;

SELECT COUNT(\*) FROM sales\_order where total = 20000;

--Сессия 1--

COMMIT;

--Сессия 2--

COMMIT;