|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №2**

**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**

1. **Работа с Rollback**

Начните транзакцию (командой BEGIN) и создайте новый заказ в таблице sales\_order с сегодняшней датой. Добавьте два предмета в таблицу item, связанных с созданным заказом.

Представьте, что пользователь не подтвердил заказ и все введённые данные необходимо отменить. Выполните отмену транзакции и проверьте, что никакой добавленной вами информации действительно не осталось.

Теперь представьте сценарий, в котором нужно отменить не все данные, а только последний из добавленных предметов. Для этого повторите все действия из предыдущего упражнения, но перед добавлением каждого предмета создавайте точку сохранения (с одним и тем же именем). После ввода второго предмета выполните откат к точке сохранения. Проверьте, что заказ и первый предмет остались.

В рамках той же транзакции добавьте еще один предмет и зафиксируйте транзакцию. Обратите внимание на то, что после этой операции отменить внесённые транзакцией изменения будет уже невозможно.

1. **Уровень изоляции Read Committed**

Перед началом выполнения задания проверьте, что в таблице sales\_order нет заказов на сумму total 1 000 рублей.

* В первом сеансе начните транзакцию (командой BEGIN). Выполните обновление таблицы sales\_order: увеличьте total в два раза в тех строках, где сумма равна 1 000 рублей.
* Во втором сеансе (откройте новое окно psql). Начните транзакцию (командой BEGIN). Вставьте в таблицу sales\_order новый заказ на 1 000 рублей и зафиксируйте транзакцию.
* В первом сеансе повторите обновление таблицы sales\_order и зафиксируйте транзакцию.

Осталась ли сумма добавленного заказа равной 1 000 рублей? Почему это не так?

1. **Уровень изоляции Repeatable Read**

Повторите предыдущее упражнение, но начните транзакцию в первом сеансе с уровнем изоляции транзакций Repeatable Read. Объясните различие полученных результатов.

Выполните указанные действия в двух сеансах:

* В первом сеансе начните новую транзакцию с уровнем изоляции Repeatable Read. Вычислите количество заказов с суммой 20 000 рублей.
* Во втором сеансе начните новую транзакцию с уровнем изоляции Repeatable Read. Вычислите количество заказов с суммой 30 000 рублей.
* В первом сеансе добавьте новый заказ на 30 000 рублей и снова вычислите количество заказов с суммой 20 000 рублей.
* Во втором сеансе добавьте новый заказ на 20 000 рублей и снова вычислите количество заказов с суммой 30 000 рублей.
* Зафиксируйте транзакции в обоих сеансах.

Соответствует ли результат ожиданиями? Можно ли сериализовать эти транзакции (иными словами, можно ли представить такой порядок последовательного выполнения этих транзакций, при котором результат совпадает с тем, что получился при параллельном выполнении)?

**РЕФЕРАТ**

Отчет 14 страниц, 3 рисунка, 5 источников, 1 приложение.

POSTGRESQL, РАЗРАБОТКА БАЗ ДАННЫХ, СОЗДАНИЕ ТАБЛИЦ, ЗАПОЛНЕНИЕ ТАБЛИЦ, SQL

Объектом разработки является база данных.

Цель работы – разработка и заполнение базы данных.

В процессе работы проводилось практическое использование базы данных в виде создания и заполнения различных таблиц.

Результатом является подготовленное рабочее место с установленным PostgreSQL и заполненная база данных.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc7781)

[1. РАЗРАБОТКА БАЗЫ ДАННЫХ 8](#_Toc14719)

[ЗАКЛЮЧЕНИЕ 10](#_Toc11659)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc18253)

[ПРИЛОЖЕНИЕ А 12](#_Toc1669)

[ФРАГМЕНТЫ КОДА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ 12](#_Toc6355)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

БД – база данных;

СУБД – система управления базами данных.

**ВВЕДЕНИЕ**

В современной сфере баз данных, PostgreSQL занимает выдающуюся позицию, предоставляя мощный инструмент для управления информацией. Эта система управления базами данных предлагает механизм транзакций, который играет ключевую роль в обеспечении последовательности и консистентности изменений в базе данных.

Транзакции в PostgreSQL выполняют множество задач, включая поддержку атомарности, согласованности, изолированности и долговечности (ACID). Эти принципы являются фундаментальными для обеспечения надёжности и предсказуемости работы с данными. В данном контексте, изучение и понимание принципов транзакций в PostgreSQL становится необходимым для разработчиков и администраторов баз данных, стремящихся обеспечить эффективное управление информацией в сложных и динамичных проектах.

Целью данного проекта является практическое исследование аспектов работы с транзакциями в базах данных на примере системы управления PostgreSQL. Проект сфокусирован на сценариях отмены транзакций, рассматривается уровни изоляции и их влияние на консистентность данных. Работа предоставляет конкретные примеры, демонстрируя важность правильного управления транзакциями в реальных сценариях.

1. **РАЗРАБОТКА БАЗЫ ДАННЫХ**

## Создание базы данных

Для работы с базой данных использовалась СУБД PostgreSQL. Результат создания базы данных представлен на рисунке 1.



Рисунок 1 – Результат создания БД

После создания базы данных и переключения на неё были созданы таблицы. Для создания таблицы использовалась конструкция “CREATE TABLE” с прописыванием названия таблицы и названия колонок с дополнительной информацией о них внутри скобок. Пример создания таблицы представлен на рисунке 2. Код создания оставшихся таблиц представлен в приложении в листинге 1.

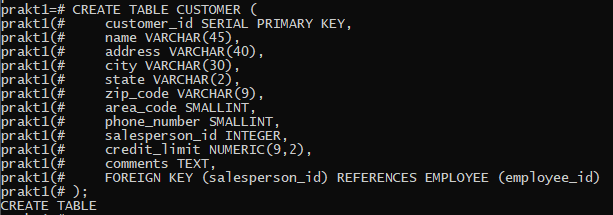


Рисунок 2 – Пример создания таблицы

## Заполнение базы данных

После создания таблиц они были заполнены записями. Для заполнения использовалась конструкция “INSERT INTO” с наименованием таблицы и перечислением колонок внутри круглых скобок, для указания значений добавлялась конструкция VALUES с перечислением данных внутри круглых скобок. Пример заполнения таблиц представлен на рисунке 3. Код вставки данных в оставшиеся таблицы представлен в приложении в листинге 2.

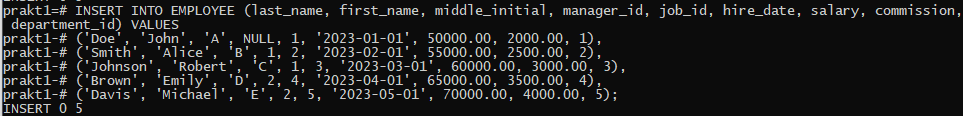


Рисунок 3 – Пример заполнения таблицы

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы было подготовлено рабочее место, с помощью установки СУБД PostgreSQL версии 15. Также была создана база данных, в которой было добавлено 9 таблиц с колонками различных типов и названий, предоставленными в задании. После создания данные таблицы были заполнены различными данными. По результату каждая из 9 таблиц, созданных по представленной в задании схеме, содержала по 5 записей с различными данными, в соответсвии с представленными типами 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. PostgreSQL: Primary Key Constraint. [Электронный ресурс] – URL: https://www.tutorialsteacher.com/postgresql/foreign-key (дата обращения: 14.02.2024).
5. PostgreSQL Foreign Key Constraint. [Электронный ресурс] – URL: https://www.tutorialsteacher.com/postgresql/foreign-key (дата обращения: 17.12.2023)

# ПРИЛОЖЕНИЕ А

**ФРАГМЕНТЫ КОДА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ**

Листинг 1 – Фрагмент кода создания таблиц

CREATE TABLE LOCATION (

location\_id SERIAL PRIMARY KEY,

regional\_group VARCHAR(20)

);

CREATE TABLE DEPARTMENT (

department\_id SERIAL PRIMARY KEY,

name VARCHAR(14),

location\_id INTEGER,

FOREIGN KEY (location\_id) REFERENCES LOCATION (location\_id)

);

CREATE TABLE JOB (

job\_id SERIAL PRIMARY KEY,

function VARCHAR(30)

);

CREATE TABLE EMPLOYEE (

employee\_id SERIAL PRIMARY KEY,

last\_name VARCHAR(15),

first\_name VARCHAR(15),

middle\_initial VARCHAR(1),

manager\_id INTEGER,

CONSTRAINT ref\_emp\_manager FOREIGN KEY (manager\_id) REFERENCES EMPLOYEE(employee\_id),

job\_id INTEGER,

hire\_date DATE,

salary NUMERIC(7,2),

commission NUMERIC(7,2),

department\_id INTEGER,

FOREIGN KEY (job\_id) REFERENCES JOB (job\_id),

FOREIGN KEY (department\_id) REFERENCES DEPARTMENT (department\_id)

);

Продолжение листинга 1

CREATE TABLE CUSTOMER (

    customer\_id SERIAL PRIMARY KEY,

    name VARCHAR(45),

    address VARCHAR(40),

    city VARCHAR(30),

    state VARCHAR(2),

    zip\_code VARCHAR(9),

    area\_code SMALLINT,

    phone\_number SMALLINT,

    salesperson\_id INTEGER,

    credit\_limit NUMERIC(9,2),

    comments TEXT,

    FOREIGN KEY (salesperson\_id) REFERENCES EMPLOYEE (employee\_id)

);

CREATE TABLE SALES\_ORDER (

    order\_id SERIAL PRIMARY KEY,

    order\_date DATE,

    customer\_id INTEGER,

    ship\_date DATE,

    total NUMERIC(8,2),

    FOREIGN KEY (customer\_id) REFERENCES CUSTOMER (customer\_id)

);

CREATE TABLE PRODUCT (

    product\_id SERIAL PRIMARY KEY,

    description VARCHAR(30)

);

Продолжение листинга 1

CREATE TABLE ITEM (

item\_id SERIAL,

order\_id INTEGER,

product\_id INTEGER,

actual\_price NUMERIC(8,2),

quantity INTEGER,

total NUMERIC(8,2),

PRIMARY KEY (item\_id, order\_id),

FOREIGN KEY (order\_id) REFERENCES SALES\_ORDER (order\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT (product\_id)

);

CREATE TABLE PRICE (

product\_id INTEGER,

start\_date DATE,

list\_price NUMERIC(8,2),

min\_price NUMERIC(8,2),

end\_date DATE,

PRIMARY KEY (product\_id, start\_date),

FOREIGN KEY (product\_id) REFERENCES PRODUCT (product\_id)

);

Листинг 2 – Фрагмент кода заполнения базы данных

INSERT INTO LOCATION (regional\_group) VALUES

('Area A'),

('Area B'),

('Area C'),

('Area D'),

('Area E');

INSERT INTO DEPARTMENT (name, location\_id) VALUES

('Department 1', 1),

('Department 2', 2),

('Department 3', 3),

('Department 4', 4),

('Department 5', 5);

Продолжение листинга 2

INSERT INTO JOB (function) VALUES

('Function 1'),

('Function 2'),

('Function 3'),

('Function 4'),

('Function 5');

INSERT INTO EMPLOYEE (last\_name, first\_name, middle\_initial, manager\_id, job\_id, hire\_date, salary, commission, department\_id) VALUES

('Doe', 'John', 'A', NULL, 1, '2023-01-01', 50000.00, 2000.00, 1),

('Smith', 'Alice', 'B', 1, 2, '2023-02-01', 55000.00, 2500.00, 2),

('Johnson', 'Robert', 'C', 1, 3, '2023-03-01', 60000.00, 3000.00, 3),

('Brown', 'Emily', 'D', 2, 4, '2023-04-01', 65000.00, 3500.00, 4),

('Davis', 'Michael', 'E', 2, 5, '2023-05-01', 70000.00, 4000.00, 5);

INSERT INTO CUSTOMER (name, address, city, state, zip\_code, area\_code, phone\_number, salesperson\_id, credit\_limit) VALUES

('Customer 1', '123 Main St', 'City A', 'CA', '12345', 123, 4567890, 1, 10000.00),

('Customer 2', '456 Elm St', 'City B', 'NY', '54321', 456, 7890123, 2, 15000.00),

('Customer 3', '789 Oak St', 'City C', 'TX', '67890', 789, 1234567, 3, 20000.00),

('Customer 4', '101 Pine St', 'City D', 'FL', '09876', 234, 5678901, 4, 25000.00),

('Customer 5', '202 Maple St', 'City E', 'WA', '56789', 345, 6789012, 5, 30000.00);

Продолжение листинга 2

INSERT INTO SALES\_ORDER (order\_date, customer\_id, ship\_date, total) VALUES

('2023-01-01', 1, '2023-01-05', 1000.00),

('2023-02-01', 2, '2023-02-05', 1500.00),

('2023-03-01', 3, '2023-03-05', 2000.00),

('2023-04-01', 4, '2023-04-05', 2500.00),

('2023-05-01', 5, '2023-05-05', 3000.00);

INSERT INTO PRODUCT (description) VALUES

('Product 1'),

('Product 2'),

('Product 3'),

('Product 4'),

('Product 5');

INSERT INTO ITEM (order\_id, product\_id, actual\_price, quantity, total) VALUES

(1, 1, 10.00, 1, 10.00),

(2, 2, 20.00, 2, 40.00),

(3, 3, 30.00, 3, 90.00),

(4, 4, 40.00, 4, 160.00),

(5, 5, 50.00, 5, 250.00);

INSERT INTO PRICE (product\_id, start\_date, list\_price, min\_price, end\_date) VALUES

(1, '2024-01-01', 10.00, 8.00, '2024-12-31'),

(2, '2024-01-01', 20.00, 16.00, '2024-12-31'),

(3, '2024-01-01', 30.00, 24.00, '2024-12-31'),

(4, '2024-01-01', 40.00, 32.00, '2024-12-31'),

(5, '2024-01-01', 50.00, 40.00, '2024-12-31');