МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2
по дисциплине
«Распределённые системы хранения данных»

Вариант №1745

Выполнил:
Студент группы P3334
Баянов Равиль
Динарович
Преподаватель:
Николаев Владимир
Вячеславович

Санкт-Петербург
2025

Содержание

Задание

Цель работы - на выделенном узле создать и сконфигурировать новый кластер БД Postgres, саму БД, табличные пространства и новую роль, а также произвести наполнение базы в соответствии с заданием. Отчёт по работе должен содержать все команды по настройке, скрипты, а также измененные строки конфигурационных файлов.

Номер выделенного узла: pg108
Логин: postgres0
Пароль: DLvc7uCc

Описание этапов выполнения

Этап 1. Инициализация кластера БД

Задание:

- Директория кластера: $HOME/zeb22
- Кодировка: ISO_8859_5
- Локаль: русская
- Параметры инициализации задать через переменные окружения

Выполнение:

```
mkdir zeb22      # Создание директории, в которой будет располагаться кластер
# Задание переменных окружения для конфигурации кодировки и локали кластера
export PGDATA=$HOME/zeb22 # Указание директории для кластера
export LANG=ISO_8859_5
export LC_COLLATE=ISO_8859_5
export LC_CTYPE=ISO_8859_5
export LC_ALL=ISO_8859_5


initdb # Инициализация кластера (все переменыне окружения применятся для этой утилиты)
```

Этап 2. Конфигурация и запуск сервера БД

Задание:

- Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла
- Номер порта: 9745
- Способ аутентификации TCP/IP клиентов: по паролю в открытом виде
- Остальные способы подключений запретить.
- Настроить следующие параметры сервера БД:
  - max_connections
  - shared_buffers
  - temp_buffers
  - work_mem
  - checkpoint_timeout
  - effective_cache_size
  - fsync
  - commit_delay

  Параметры должны быть подобраны в соответствии со сценарием OLTP: 300 одновременных пользователей, 2 сессий на каждого; каждая сессия инициирует до 7 транзакций на запись размером 8КБ; обеспечить максимальную производительность.

- Директория WAL файлов: $PGDATA/pg_wal
- Формат лог-файлов: .log
- Уровень сообщений лога: INFO
- Дополнительно логировать: контрольные точки и попытки подключения

Выполнение:

```
# Изменения в файле postgres.conf

listen_adresses = '*' # Принимаем подключения по любом IP-адресу
port = 9745
max_connections = 600 # 2 * 300 (кол-во соединений для кластера)
shared_buffers = 1GB # Размер кэша для страниц баз данных (20-30% от RAM для OLTP)
temp_buffers = 32MB # Размер памяти для временных таблиц одной сессии
work_mem = 8MB # Размер памяти для различных операций в рамках одного запроса
fsync = on # Гарантия того, что данные сохранятся на диск перед завершением транзакции, теряется скорость, но в любом случае данные мы терять не хотим
commit_delay = 1000 # 1 мс - время ожидания перед группированием операций в одну транзакцию для передачи на диск (уменьшаем число дисковых операций)
log_destination = 'stderr' # Перенаправляем лог ошибок в стандартный поток вывода ошибок для перемещения его в файл с логами
log_directory = 'pg_log' # Директория, в которой будут храниться логи
```

```
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # паттерн названия файла с логами (указание даты создания)
logging_collector = on # Включаем перенаправление потока stderr в файл
log_file_mode = 0740 # Указываем права для лог файлов
log_rotation_age = 1d # Время через которое логи очищаются и перезаписываются
log_min_message = info # Минимальный уровень сообщений, записывающихся в логи
log_chekpoints = on # Логирование информации о записи данных на диск
log_connections = on
log_disconnection = on

# Изменения в файле pg_hba.conf
local all all    peer # Unix-domain сокет в режиме peer
host all all 0.0.0.0/0 password # пособ аутентификации TCP/IP клиентов: по паролю в открытом виде для IPv4 для всех IP-адресов
host all all ::0/0 password # для IPv6

# Всё остальное закомментим, чтобы исключить другие виды подключений
```

Логирование:

```
[postgres0@pg108 ~/zeb22/pg_log]$ psql -p 9745
psql (16.4)
Введите "help", чтобы получить справку.

postgres0=# exit
[postgres0@pg108 ~/zeb22/pg_log]$ cat postgresql-2025-03-10_034837.log
2025-03-10 03:48:37.580 MSK [75645] СООБЩЕНИЕ:  запускается PostgreSQL 16.4 on amd64-portbld-freebsd14.1, compil
ed by FreeBSD clang version 18.1.6 (https://github.com/llvm/llvm-project.git llvmorg-18.1.6-0-g1118c2e05e67), 64
-bit
2025-03-10 03:48:37.580 MSK [75645] СООБЩЕНИЕ:  для приёма подключений по адресу IPv6 "::" открыт порт 9745
2025-03-10 03:48:37.580 MSK [75645] СООБЩЕНИЕ:  для приёма подключений по адресу IPv4 "0.0.0.0" открыт порт 9745
2025-03-10 03:48:37.593 MSK [75645] СООБЩЕНИЕ:  для приёма подключений открыт Unix-сокет "/tmp/.s.PGSQL.9745"
2025-03-10 03:48:37.662 MSK [75649] СООБЩЕНИЕ:  система БД была выключена: 2025-03-10 03:48:37 MSK
2025-03-10 03:48:37.694 MSK [75645] СООБЩЕНИЕ:  система БД готова принимать подключения
2025-03-10 03:49:16.953 MSK [75679] СООБЩЕНИЕ:  принято подключение: узел=[local]
2025-03-10 03:49:16.980 MSK [75679] СООБЩЕНИЕ:  соединение аутентифицировано: идентификатор="postgres0" метод=pe
er (/var/db/postgres0/zeb22/pg_hba.conf:117)
2025-03-10 03:49:16.980 MSK [75679] СООБЩЕНИЕ:  подключение авторизовано: пользователь=postgres0 база=postgres0
приложение=psql
2025-03-10 03:49:19.165 MSK [75679] СООБЩЕНИЕ:  ◆◆◆◆◆◆◆◆◆◆: ◆◆◆◆◆ ◆◆◆◆◆◆: 0:00:02.211 ◆◆◆◆◆◆◆◆◆◆◆◆=postgres0 ◆◆◆
◆ ◆◆◆◆◆◆=postgres0 ◆◆◆◆◆◆◆◆◆=[local]
```

Этап 3. Дополнительные табличные пространства и наполнение базы

Задание:

- Пересоздать шаблон template1 в новом табличном пространстве: $HOME/rez82
- На основе template0 создать новую базу: nicebluemom
- Создать новую роль, предоставить необходимые права, разрешить подключение к базе.
- От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ табличные пространства должны использоваться по назначению.
- Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Выполнение:

```
mkdir rez82 # Создание директории для нового табличного пространства

# После подключения к самой БД
CREATE TABLESPACE rez82 LOCATION '/var/db/postgres0/rez82'; # Создание
табличного пространства
UPDATE pg_database
SET dattablespace = (SELECT oid FROM pg_tablespace WHERE spcname = 'rez82')
WHERE datname = 'template1'; # Перемещение шаблона template1 в пространство
rez82
CREATE DATABASE nicebluemom WITH TEMPLATE template0 TABLESPACE rez82; #
Создание БД nicebluemom в пространстве rez82

CREATE ROLE test_role WITH LOGIN PASSWORD 'test_role'; # Создание роли
 ALTER ROLE test_role CREATEDB CREATEROLE; # Добавление возможности создания
БД и ролей для новой роли
GRANT CONNECT ON DATABASE postgres TO test_role; Разрешение подключаться к БД
postgres
GRANT CONNECT ON DATABASE postgres0 TO test_role; Разрешение подключения к
postgres0
GRANT CONNECT ON DATABASE nicebluemom TO test_role; Разрешение подключения к
nicebluemom
# Разрешение манипулировать данными в БД для новой тестовой роли
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO test_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO
test_role;
SET ROLE test_role; # Смена роли
```

Теперь создадим скрипт заполнения всех пространств кластера. Представим, что:
nicebluemom – это сервис для наставничества (менторства) для студентов.
postgres - Главная база пользователей
nicebluemom - База курсов и наставничества
postgres0 - База транзакционных данных (оценки, регистрация, взаимодействие)

Скрипт:

```
-- Переключаемся на базу данных nuicebluemom
\c nicebluemom;

-- Убедитесь, что таблицы создаются в нужном табличном пространстве
SET default_tablespace = 'rez82';

-- Таблица клиентов
CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15),
    address TEXT
);

-- Таблица продуктов
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    stock_quantity INT NOT NULL
);

-- Таблица заказов
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES customers(customer_id) ON DELETE CASCADE,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) CHECK (status IN ('pending', 'shipped', 'delivered',
'canceled'))
);

-- Таблица позиций заказа (каждый заказ может включать несколько продуктов)
CREATE TABLE order_items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(order_id) ON DELETE CASCADE,
    product_id INT REFERENCES products(product_id) ON DELETE CASCADE,
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

-- Таблица транзакций (для отслеживания платежей за заказы)
CREATE TABLE transactions (
    transaction_id SERIAL PRIMARY KEY,
    order_id INT REFERENCES orders(order_id) ON DELETE CASCADE,
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    amount DECIMAL(10, 2) NOT NULL,
    status VARCHAR(20) CHECK (status IN ('pending', 'completed', 'failed'))
);

-- Индексы для улучшения производительности
CREATE INDEX idx_order_customer ON orders(customer_id);
CREATE INDEX idx_product_name ON products(name);
```

```sql
-- Вставка клиента
INSERT INTO customers (first_name, last_name, email, phone, address)
VALUES ('John', 'Doe', 'john.doe@example.com', '123-456-7890', '1234 Elm St, Springfield');

-- Вставка продукта
INSERT INTO products (name, description, price, stock_quantity)
VALUES ('Smartphone', 'Latest model of smartphone', 599.99, 100);

-- Создание заказа
INSERT INTO orders (customer_id, status)
VALUES (1, 'pending');

-- Вставка позиции заказа
INSERT INTO order_items (order_id, product_id, quantity, price)
VALUES (1, 1, 1, 599.99);

-- Вставка транзакции
INSERT INTO transactions (order_id, amount, status)
VALUES (1, 599.99, 'pending');









-- Переключаемся на базу данных postgres0
\c postgres0;

-- Таблица поставщиков
CREATE TABLE suppliers (
    supplier_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    contact_name VARCHAR(100),
    contact_phone VARCHAR(15),
    address TEXT
);

-- Вставка поставщика
INSERT INTO suppliers (name, contact_name, contact_phone, address)
VALUES ('Tech Corp', 'Alice Smith', '987-654-3210', '7890 Tech Lane, Silicon Valley');
```

Вывод все объектов:

```sql
WITH spaced_tables AS (
SELECT
COALESCE(t.spcname, 'pg_default') AS spcname,
c.relname,
ROW_NUMBER() OVER (PARTITION BY COALESCE(t.spcname, 'pg_default') ORDER BY c.relname) AS rn
FROM pg_tablespace t
FULL JOIN pg_class c ON c.reltablespace = t.oid
ORDER BY spcname, c.relname
)
SELECT
```

```
CASE WHEN rn = 1 THEN spcname ELSE NULL END AS spcname,
relname
FROM spaced_tables;
```

Сложности

При выполнении данной лабораторной работы возникло не мало сложностей. Первый нюанс, с которым я столкнулся – это то, что совсем неясно было какой оперативной памятью обладает узел, на котором я работаю, следовательно были проблемы с конфигурацией кластера так, чтобы подключение и логирование работали корректно и без ошибок. Приходилось подстраивать размеры кэша и разбираться детально в том, как работает поле logging_collectors в конфигурационном файле postgres.conf. Также были сложности в наполнении кластера, так как это отняло много времени и зачастую мои скрипты не работали правильно. Также к этому не сразу удалось установить зависимость параметров кластера БД с OLTP подходом. Но все трудности были устранены.

Вывод

Выполнив данную лабораторную работу, я создал и сконфигурировал свой кластер баз данных, научился настраивать базу данных для любых нужд. Узнал, как можно регулировать подключение к базе данных в зависимости от ролей, табличных пространств и настроек подключения. Также у меня получилось наполнить базу данных по технологии OLTP и теперь в моих силах настраивать кластеры PostgreSQL под любую задачу.