Федеральное государственное автономное образовательное

учреждение высшего образования

Университет ИТМО

Распределённые системы хранения данных **Лабораторная работа 2**

Вариант 85426, pg121, пользователь postgres2

**Выполнил:**

Митичев Иван Дмитриевич

**Группа:**

P3316

**Преподаватель:**

Николаев Владимир Вячеславович

2025 г.

Санкт-Петербург

# 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.

Способ подключения к узлу из сети Интернет через helios:  
ssh -J sXXXXXX@helios.cs.ifmo.ru:2222 postgresY@pgZZZ  
Способ подключения к узлу из сети факультета:  
ssh postgresY@pgZZZ  
Номер выделенного узла pgZZZ, а также логин и пароль для подключения Вам выдаст преподаватель.

Этап 1. Инициализация кластера БД

* Директория кластера: $HOME/ykk83
* Кодировка: ANSI1251
* Локаль: русская
* Параметры инициализации задать через аргументы команды

Этап 2. Конфигурация и запуск сервера БД

* Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, только localhost
* Номер порта: 9426
* Способ аутентификации TCP/IP клиентов: по имени пользователя
* 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:  
250 одновременных пользователей, 3 сессий на каждого; каждая сессия инициирует до 10 транзакций на запись размером 24КБ; обеспечить максимальную производительность.

* Директория WAL файлов: $PGDATA/qfg95
* Формат лог-файлов: .csv
* Уровень сообщений лога: ERROR
* Дополнительно логировать: контрольные точки и попытки подключения

Этап 3. Дополнительные табличные пространства и наполнение базы

* На основе шаблона template0 пересоздать базу postgres в новом табличном пространстве: $HOME/twv39
* На основе template1 создать новую базу: lazywhitewood
* 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.

# Подключение

Способ подключения к узлу из сети Интернет через helios:  
ssh -J s368527@helios.cs.ifmo.ru:2222 postgres2@pg121  
Способ подключения к узлу из сети факультета:

ssh -p 2222 [s368527@se.ifmo.ru](mailto:s368527@se.ifmo.ru)  
ssh postgres2@pg121

# Инициализация кластера БД

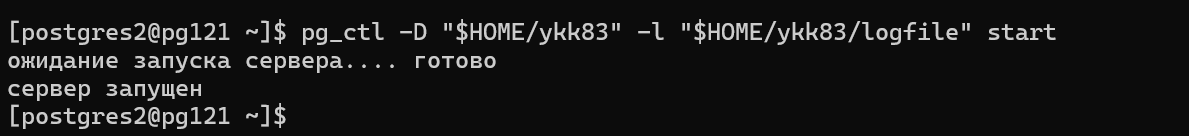
* Директория кластера: $HOME/ykk83
* Кодировка: ANSI1251
* Локаль: русская
* Параметры инициализации задать через аргументы команды

ssh -J s368527@helios.cs.ifmo.ru:2222 postgres2@pg121  
PGDATA=$HOME/ykk83  
export PGDATA  
# Создание каталога под кластер  
mkdir -p $HOME/ykk83  
#Создаём новую директорию под WAL  
mkdir -p $HOME/qfg95  
# Инициализация кластера (аргументы команды initdb)  
initdb -D "$HOME/ykk83" -X "$HOME/qfg95" --locale=ru\_RU.CP1251 --encoding=WIN1251 --username=postgres2

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.

pg\_ctl -D "$HOME/ykk83" -l "$HOME/ykk83/logfile" start



# Конфигурация и запуск сервера БД

* Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, только localhost
* Номер порта: 9426
* Способ аутентификации TCP/IP клиентов: по имени пользователя
* Остальные способы подключений запретить.

postgresql.conf: Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

pg\_hba.conf:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Параметры должны быть подобраны в соответствии со сценарием OLTP:  
250 одновременных пользователей, 3 сессий на каждого; каждая сессия инициирует до 10 транзакций на запись размером 24КБ; обеспечить максимальную производительность.

**max\_connections:**

Поскольку всего планируется поддерживать 150 пользователей \* 4 сессии = 600 сессий одновременно, я выбрал значение max\_connections равным 600.



**shared\_buffers:**

Для shared\_buffers рекомендуется использовать 25% от основной памяти сервера (в случаях, когда память сервера больше 1Гб). Возьмем 2Гб.

**temp\_buffers:**

Поскольку в сценарии не планируется использование транзакций, работающих с большими объемами данных 128 MB — хорошо для OLTP с временными операциями.

**work\_mem:**

Точно не известно, на сколько часто будут использоваться запросы с сортировками и хэшированием в транзакциях, однако, лучше взять с небольшим запасом, то есть следующее значение после значения по умолчанию – 16Мб.

**checkpoint\_timeout:**

Так как в сценарии требуется обеспечить максимальную производительность, а увеличение checkpoint\_timeout приведет к нагрузке на систему из-за более частых сохранений, оставим значение по умолчанию – 5min.

**effective\_cache\_size:**

Для указанной в сценарии высокой производительности выберем effective\_cache\_size = 10Гб, чтобы больше запросов использовали индексы и выполнялись быстрее.

**fsync:**

Так как сценарий требует обеспечить максимальную производительность, отключаем fsync. Это ускорит обработку запросов, ведь теперь они не будут принудительно писаться на диск после коммита, то есть увеличится производительность, поскольку запись на диск – небыстрая операция. Однако, в случае сбоя системы, данные окажутся утеряны. Это осознанный риск, ведь про сохранность данных в сценарии ничего не сказано и требование заключается именно в максимальной производительности любыми способами.

**commit\_delay:**

Этот параметр устанавливает задержку перед сохранением в WAL. Таким образом, можно за одно сохранение зафиксировать сразу множество транзакций, увеличив таким образом производительность системы, что и нужно по сценарию. Поставим commit\_delay = 10мс.

**Директория WAL файлов: $PGDATA/qfg95**

Задали при инициализации

**Формат лог-файлов: .csv**

**Уровень сообщений лога: ERROR**

**Дополнительно логировать: контрольные точки и попытки подключения**

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

# Дополнительные табличные пространства и наполнение базы

На основе шаблона template0 пересоздать базу postgres в новом табличном пространстве: $HOME/twv39

На основе template1 создать новую базу: lazywhitewood

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.

# 1. Создаём новое табличное пространство  
mkdir -p $HOME/twv39  
  
# Подключаемся к psql как суперпользователь  
psql -p 9426 -U postgres2 -d postgres  
  
-- Создаём tablespace в /var/db/postgres2/twv39 (путь должен быть абсолютным и доступен PostgreSQL серверу)  
CREATE TABLESPACE twv39 LOCATION '/var/db/postgres2/twv39';  
#Подключаемся к другой базе (например, template1), т.к. нельзя удалить базу, к которой мы подключёны  
psql -p 9426 -U postgres2 -d template1  
-- Отключаем подключения к базе postgres (чтобы можно было её пересоздать)  
SELECT pid, usename, application\_name, client\_addr  
FROM pg\_stat\_activity  
WHERE datname = 'postgres';  
-- сюда всталвяем свой pid  
SELECT *pg\_terminate\_backend*(<pid>);  
-- Удаляем старую базу postgres  
DROP DATABASE postgres;  
-- Создаём новую базу postgres на основе template0 в tablespace twv39  
CREATE DATABASE postgres  
 TEMPLATE template0  
 TABLESPACE twv39;  
  
#Проверка  
psql -p 9426 -U postgres2 -d postgres  
--Проверяем, что она использует нужное табличное пространство:  
SELECT datname, pg\_tablespace.spcname AS tablespace  
FROM pg\_database  
 JOIN pg\_tablespace ON pg\_database.dattablespace = pg\_tablespace.oid  
WHERE datname = 'postgres';  
  
  
  
  
# Подключаемся к psql как суперпользователь  
psql -p 9426 -U postgres2 -d postgres  
  
-- Создаём новую базу lazywhitewood на основе template1  
CREATE DATABASE lazywhitewood  
 WITH TEMPLATE = template1  
 OWNER = postgres2;  
  
-- Создаём новую роль new\_role с паролем и правами подключения  
CREATE ROLE new\_role LOGIN PASSWORD 'new';  
  
-- Даем роли права подключаться к обеим базам  
GRANT CONNECT ON DATABASE postgres TO new\_role;  
GRANT CONNECT ON DATABASE lazywhitewood TO new\_role;  
  
-- Разрешаем роли создавать объекты в tablespace twv39 (если нужно)  
GRANT CREATE ON TABLESPACE twv39 TO new\_role;  
-- Создать таблицу в таблицпространстве postgres  
CREATE TABLE test\_table\_postgres (  
 id SERIAL PRIMARY KEY,  
 name TEXT NOT NULL,  
 value INTEGER  
) TABLESPACE twv39;  
  
-- Дать права на вставку и использование sequence роли new\_role  
GRANT INSERT, SELECT ON test\_table\_postgres TO new\_role;  
GRANT USAGE, SELECT ON SEQUENCE test\_table\_postgres\_id\_seq TO new\_role;  
  
-- Подключиться к базе postgres  
psql -h localhost -p 9426 -U new\_role -d postgres  
INSERT INTO test\_table\_postgres (name, value) VALUES  
 ('A', 1),  
 ('B', 2),  
 ('C', 3);  
--проверка  
SELECT \* FROM test\_table\_lw;  
  
-- Подключиться к базе lazywhitewood (под postgres2)  
psql -p 9426 -U postgres2 -d lazywhitewood  
  
-- Создать таблицу в таблицпространстве twv39  
CREATE TABLE test\_table\_lw (  
 id SERIAL PRIMARY KEY,  
 name TEXT NOT NULL,  
 value INTEGER  
) TABLESPACE twv39;  
  
-- Дать права на вставку и использование sequence роли new\_role  
GRANT INSERT, SELECT ON test\_table\_lw TO new\_role;  
GRANT USAGE, SELECT ON SEQUENCE test\_table\_lw\_id\_seq TO new\_role;  
  
-- Вставляем  
psql -h localhost -p 9426 -U new\_role -d lazywhitewood  
INSERT INTO test\_table\_lw (name, value) VALUES  
 ('X', 10),  
 ('Y', 20),  
 ('Z', 30);  
--проверка  
SELECT \* FROM test\_table\_lw;  
  
  
  
--Выведем список всех табличных пространств кластера и содержащиеся в них объекты  
SELECT  
 CASE WHEN *ROW\_NUMBER*() OVER (PARTITION BY *COALESCE*(t.spcname,  
 'pg\_default') ORDER BY c.relname) = 1  
 THEN *COALESCE*(t.spcname, 'pg\_default')  
 ELSE NULL  
 END AS spcname,  
 c.relname  
FROM pg\_tablespace t  
 FULL JOIN pg\_class c ON c.reltablespace = t.oid  
ORDER BY *COALESCE*(t.spcname, 'pg\_default'), c.relname;

# Вывод

В ходе выполнения лабораторной работы был создан и сконфигурирован

кластер БД на выделенном узле, я познакомился с различными

вариантами конфигурации. Также была создана БД, новая роль, табличные

пространства и заполнение тестовыми данными.