Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

Администрирование СУБД

Лабораторная работа №2

Вариант №456555

Группа: P3324

Выполнил: Маликов Глеб Игоревич

Преподаватель:

Николаев Владимир Вячеславович

Санкт-Петербург

2024г.

# 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/khk43
* Кодировка: ISO\_8859\_5
* 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, принимать подключения к любому IP-адресу узла
* Номер порта: 9555
* 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 с аппаратной конфигурацией:  
оперативная память 4ГБ, хранение на жёстком диске (HDD).

* Директория WAL файлов: $HOME/oka84
* Формат лог-файлов: .csv
* Уровень сообщений лога: ERROR
* Дополнительно логировать: завершение сессий и продолжительность выполнения команд

## Этап 3. Дополнительные табличные пространства и наполнение базы

* Создать новые табличные пространства для временных объектов: $HOME/mqb89, $HOME/utr38
* На основе template0 создать новую базу: uglyredbird
* 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.

# Реализация

## Этап 1. Инициализация кластера БД

Создаем директорию кластера и инициализируем базу данных:

mkdir -p $PGDATA

chown $PGUSERNAME $PGDATA

initdb -D $PGDATA --encoding=$PGENCODE --locale=$PGLOCALE --username=$PGUSERNAME

Результат:

A computer screen shot of a black screen

Description automatically generated

## Этап 2. Конфигурация и запуск сервера БД

Скачиваем конфигурационные файлы:

scp postgres1@pg167:khk43/postgresql.conf ~

scp postgres1@pg167:khk43/pg\_hba.conf ~

### Настройка способов подключения

Редактируем файл postgresql.conf:

* сокет TCP/IP, принимать подключения к любому IP-адресу узла
* Номер порта: 9555

# - Connection Settings -

listen\_addresses = '\*'      # what IP address(es) to listen on;

                    # comma-separated list of addresses;

                    # defaults to 'localhost'; use '\*' for all

                    # (change requires restart)

port = 9555             # (change requires restart)

password\_encryption = md5   # scram-sha-256 or md5

Редактируем файл pg\_hba.conf:

* Unix-domain сокет в режиме peer
* Способ аутентификации TCP/IP клиентов: по имени пользователя
* Остальные способы подключений запретить. Из-за проблем ident в Гелиос, меняем на md5.

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# Разрешить локальные подключения через Unix-domain сокет с аутентификацией peer

local   all             all                                     peer

# Разрешить TCP/IP подключения со всех IP-адресов с аутентификацией по имени пользователя (ident)

host    all             all             0.0.0.0/0               md5

host    all             all             ::/0                    md5

# Запретить все остальные подключения

local   replication     all                                     reject

host    replication     all             127.0.0.1/32            reject

host    replication     all             ::1/128                 reject

### Параметры сервера БД

**max\_connections**:

max\_connections = 100

**shared\_buffers**: Ставим 1/4 от оперативной памяти согласно документации [postgresql](https://www.postgresql.org/docs/current/runtime-config-resource.html" \l "GUC-SHARED-BUFFERS" \o "https://www.postgresql.org/docs/current/runtime-config-resource.html#GUC-SHARED-BUFFERS), т.е. 1ГБ.

shared\_buffers = 1GB

**temp\_buffers**: Количество памяти, выделенной для временных таблиц на одну сессию. Учитывая максимальное количество соединений, temp\_buffers займут 100 \* 16MB = 1600MB.

temp\_buffers = 16MB

**work\_mem**: Количество памяти, выделенной для операций сортировки и хеширования на одно соединение. Не зная какого вида операции будут производиться, (сложные соединения и сортировки или простые запросы) то оставляем значение по умолчанию 4MB. Work\_mem максимально может занимать 100 \* 4MB = 400MB.

work\_mem = 4MB

**checkpoint\_timeout**: Интервал времени между контрольными точками (checkpoints). Контрольные точки обеспечивают согласованность данных на диске. Учитывая, что у нас HDD, более длинный интервал времени между контрольными точками уменьшит нагрузку на диск.

checkpoint\_timeout = 15min

**effective\_cache\_size**: Этот параметр представляет собой оценку для планировщика о количестве дискового кэша, доступного для PostgreSQL. Это значение должно быть больше shared\_buffers. Учитывая, что у нас HDD, то операции ввода-вывода будут медленными, поэтому считывание из кэша будет предпочтительнее. Так, ставим 75% от оперативной памяти, т.е. 3ГБ.

effective\_cache\_size = 3GB

**fsync**: Этот параметр должен быть включен для обеспечения безопасности данных в случае сбоя системы. Отключение этого параметра может улучшить производительность, но риск потери данных в случае сбоя неприемлем для большинства производственных систем.

fsync = on

**commit\_delay**: Этот параметр задает задержку в миллисекундах перед сохранением WAL. Без тестирования сложно подобрать оптимальное значение. По умолчанию 0.

commit\_delay = 0

### WAL файлы и логирование:

**Директория WAL файлов:**

Создадим директорию для WAL файлов:

mkdir -p $HOME/oka84

chown $PGUSERNAME $HOME/oka84

* archive\_mode - включает архивирование WAL файлов.
* archive\_command - команда, которая будет выполняться для архивирования WAL файлов. В данном случае, копируем файл в директорию $HOME/oka84.

archive\_mode = on

archive\_command = 'cp %p $HOME/oka84/%f'

**Формат лог-файлов**:

* log\_destination - куда писать логи. В данном случае, в файл csv.
* logging\_collector - включает сборщик логов и позволяет перенаправлять в файлы.
* log\_directory - директория для логов. Оставляем по умолчанию.
* log\_filename - формат имени файла лога. Ставим формат csv.

log\_destination = 'csvlog'

logging\_collector = on

log\_directory = 'log'

log\_filename = 'postgresql-%Y-%m-%d\_%H%M%S.csv'

**Уровень сообщений лога**:

* log\_min\_messages - минимальный уровень сообщений, которые будут записаны в лог. В данном случае, только ошибки и выше.

log\_min\_messages = error

**Дополнительно логировать**:

* log\_connections - логировать подключения.
* log\_disconnections - логировать отключения. Оба параметра используем для отслеживания завершения сессий.
* log\_duration - логировать продолжительность выполнения команд.
* log\_min\_duration\_statement - минимальная продолжительность выполнения команды, которая будет логироваться. В данном случае 0 - логировать все команды.

log\_connections = on

log\_disconnections = on

log\_duration = on

log\_min\_duration\_statement = 0

### Запуск сервера БД

Загрузим обратно конфигурационные файлы:

scp ~/postgresql.conf postgres1@pg167:khk43

scp ~/pg\_hba.conf postgres1@pg167:khk43

Запускаем сервер:

pg\_ctl -D /var/db/postgres1/khk43 -l файл\_журнала start

### Проверка всех параметров

Статус сервера:

[postgres1@pg167 ~]$ pg\_ctl -D ~/khk43 status

pg\_ctl: сервер работает (PID: 63080)

/usr/local/bin/postgres "-D" "/var/db/postgres1/khk43"

Подключение локально:

[postgres1@pg167 ~]$ psql -p 9555 -d postgres

psql (16.4)

Введите "help", чтобы получить справку.

postgres=#

Подключение удаленно

Создадим нового пользователя PostgreSQL с паролем:

CREATE ROLE testuser WITH LOGIN PASSWORD 'testpassword';

Попробуем подключиться удаленно:

[s372819@helios ~]$ psql -h pg167 -p 9555 -U testuser -d postgres

Пароль пользователя testuser:

psql (16.4)

Введите "help", чтобы получить справку.

postgres=>

Проверка параметров:

postgres=# SHOW max\_connections;

SHOW shared\_buffers;

SHOW temp\_buffers;

SHOW work\_mem;

SHOW checkpoint\_timeout;

SHOW effective\_cache\_size;

SHOW fsync;

SHOW commit\_delay;

 max\_connections

-----------------

 100

(1 строка)

 shared\_buffers

----------------

 1GB

(1 строка)

 temp\_buffers

--------------

 16MB

(1 строка)

 work\_mem

----------

 4MB

(1 строка)

 checkpoint\_timeout

--------------------

 15min

(1 строка)

 effective\_cache\_size

----------------------

 3GB

(1 строка)

 fsync

-------

 on

(1 строка)

 commit\_delay

--------------

 0

(1 строка)

postgres=#

## Этап 3. Дополнительные табличные пространства и наполнение базы

### Создание табличных пространств

mkdir -p /var/db/postgres1/mqb89

mkdir -p /var/db/postgres1/utr38

CREATE TABLESPACE mqb89 LOCATION '/var/db/postgres1/mqb89';

CREATE TABLESPACE utr38 LOCATION '/var/db/postgres1/utr38';

Проверка:

postgres=# \db

           Список табличных пространств

    Имя     | Владелец  |      Расположение

------------+-----------+-------------------------

 mqb89      | postgres1 | /var/db/postgres1/mqb89

 pg\_default | postgres1 |

 pg\_global  | postgres1 |

 utr38      | postgres1 | /var/db/postgres1/utr38

(4 строки)

### Создание базы данных

CREATE DATABASE uglyredbird TEMPLATE template0;

postgres=# \l

                                                                Список баз данных

     Имя     | Владелец  | Кодировка  | Провайдер локали |   LC\_COLLATE    |    LC\_CTYPE     | локаль ICU | Правила ICU |      Права доступа

-------------+-----------+------------+------------------+-----------------+-----------------+------------+-------------+-------------------------

 postgres    | postgres1 | ISO\_8859\_5 | libc             | ru\_RU.ISO8859-5 | ru\_RU.ISO8859-5 |            |             |

 template0   | postgres1 | ISO\_8859\_5 | libc             | ru\_RU.ISO8859-5 | ru\_RU.ISO8859-5 |            |             | =c/postgres1           +

             |           |            |                  |                 |                 |            |             | postgres1=CTc/postgres1

 template1   | postgres1 | ISO\_8859\_5 | libc             | ru\_RU.ISO8859-5 | ru\_RU.ISO8859-5 |            |             | =c/postgres1           +

             |           |            |                  |                 |                 |            |             | postgres1=CTc/postgres1

 uglyredbird | postgres1 | ISO\_8859\_5 | libc             | ru\_RU.ISO8859-5 | ru\_RU.ISO8859-5 |            |             |

(4 строки)

### Создание роли

CREATE ROLE newuser WITH LOGIN;  --Пароль не нужен так как используем подключение peer

-- Предоставить необходимые права

GRANT CONNECT, CREATE ON DATABASE uglyredbird TO newuser;

GRANT CREATE ON TABLESPACE mqb89 TO newuser;

GRANT CREATE ON TABLESPACE utr38 TO newuser;

### Наполнение созданных баз тестовыми наборами данных.

Запускаем скрипт наполнения базы от имени нового пользователя:

psql -p 9555 -d uglyredbird -U newuser -f $HOME/newuser.sql

Проверка:

uglyredbird=> SELECT \* FROM pg\_catalog.pg\_tables WHERE tableowner = 'newuser';

 schemaname |       tablename        | tableowner | tablespace | hasindexes | hasrules | hastriggers | rowsecurity

------------+------------------------+------------+------------+------------+----------+-------------+-------------

 main       | students               | newuser    |            | t          | f        | f           | f

 main       | courses                | newuser    |            | t          | f        | f           | f

 pg\_temp\_3  | temp\_enrollments       | newuser    | mqb89      | t          | f        | f           | f

 pg\_temp\_3  | temp\_course\_statistics | newuser    | utr38      | f          | f        | f           | f

(4 строки)

### Вывести список всех табличных пространств кластера и содержащиеся в них объекты

Выведем все табличные пространства

uglyredbird=> SELECT \* FROM pg\_tablespace;

  oid  |  spcname   | spcowner |                   spcacl                    | spcoptions

-------+------------+----------+---------------------------------------------+------------

  1663 | pg\_default |       10 |                                             |

  1664 | pg\_global  |       10 |                                             |

 16389 | mqb89      |       10 | {postgres1=C/postgres1,newuser=C/postgres1} |

 16390 | utr38      |       10 | {postgres1=C/postgres1,newuser=C/postgres1} |

(4 строки)

Выведем все объекты в табличных пространствах

SELECT

    spcname AS tablespace,

    relname

FROM

    pg\_class

    JOIN pg\_tablespace ON pg\_tablespace.oid = reltablespace;

 tablespace |                 relname

------------+-----------------------------------------

 pg\_global  | pg\_toast\_1262

 pg\_global  | pg\_toast\_1262\_index

 pg\_global  | pg\_toast\_2964

 pg\_global  | pg\_toast\_2964\_index

 pg\_global  | pg\_toast\_1213

 pg\_global  | pg\_toast\_1213\_index

 pg\_global  | pg\_toast\_1260

 pg\_global  | pg\_toast\_1260\_index

 pg\_global  | pg\_toast\_2396

 pg\_global  | pg\_toast\_2396\_index

 pg\_global  | pg\_toast\_6000

 pg\_global  | pg\_toast\_6000\_index

 pg\_global  | pg\_toast\_3592

 pg\_global  | pg\_toast\_3592\_index

 pg\_global  | pg\_toast\_6243

 pg\_global  | pg\_toast\_6243\_index

 pg\_global  | pg\_toast\_6100

 pg\_global  | pg\_toast\_6100\_index

 pg\_global  | pg\_database\_datname\_index

 pg\_global  | pg\_database\_oid\_index

 pg\_global  | pg\_db\_role\_setting\_databaseid\_rol\_index

 pg\_global  | pg\_tablespace\_oid\_index

 pg\_global  | pg\_tablespace\_spcname\_index

 pg\_global  | pg\_authid\_rolname\_index

 pg\_global  | pg\_authid\_oid\_index

 pg\_global  | pg\_auth\_members\_oid\_index

 pg\_global  | pg\_auth\_members\_role\_member\_index

 pg\_global  | pg\_auth\_members\_member\_role\_index

 pg\_global  | pg\_auth\_members\_grantor\_index

 pg\_global  | pg\_shdepend\_depender\_index

 pg\_global  | pg\_shdepend\_reference\_index

 pg\_global  | pg\_shdescription\_o\_c\_index

 pg\_global  | pg\_replication\_origin\_roiident\_index

 pg\_global  | pg\_replication\_origin\_roname\_index

 pg\_global  | pg\_shseclabel\_object\_index

 pg\_global  | pg\_parameter\_acl\_parname\_index

 pg\_global  | pg\_parameter\_acl\_oid\_index

 pg\_global  | pg\_subscription\_oid\_index

 pg\_global  | pg\_subscription\_subname\_index

 pg\_global  | pg\_authid

 mqb89      | temp\_enrollments

 utr38      | temp\_course\_statistics

 pg\_global  | pg\_subscription

 pg\_global  | pg\_database

 pg\_global  | pg\_db\_role\_setting

 pg\_global  | pg\_tablespace

 pg\_global  | pg\_auth\_members

 pg\_global  | pg\_shdepend

 pg\_global  | pg\_shdescription

 pg\_global  | pg\_replication\_origin

 pg\_global  | pg\_shseclabel

 pg\_global  | pg\_parameter\_acl

(52 строки)

Выведем все объекты, созданные новым пользователем:

SELECT

    relname, spcname AS tablespace

FROM

    pg\_class LEFT JOIN pg\_tablespace ON pg\_tablespace.oid = reltablespace

WHERE

    relowner = (SELECT oid FROM pg\_roles WHERE rolname = 'newuser');

           relname            | tablespace

------------------------------+------------

 students\_student\_id\_seq      |

 students                     |

 students\_pkey                |

 courses\_course\_id\_seq        |

 courses                      |

 courses\_pkey                 |

 temp\_enrollments\_temp\_id\_seq |

 temp\_enrollments             | mqb89

 temp\_enrollments\_pkey        |

 temp\_course\_statistics       | utr38

(10 строк)

# Вывод

В результате выполнения лабораторной работы были успешно достигнуты все поставленные цели, связанные с созданием и конфигурированием нового кластера базы данных PostgreSQL на выделенном узле. Работа включала три основных этапа: инициализацию кластера БД, конфигурацию и запуск сервера БД, а также создание дополнительных табличных пространств и наполнение базы тестовыми данными. Лабораторная работа позволила углубить знания и приобрести практические навыки в настройке и управлении кластером базы данных PostgreSQL.