Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 2 по РСХД

Кластер PostgreSQL

Вариант 39455

Группа: P3316

Выполнили:

Сиразетдинов, Шпинева

Проверил:

Николаев В.В.

г. Санкт-Петербург

2025

Задание

Цель работы

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/djs10`
- Кодировка: ANSI1251
- Локаль: русская

Параметры инициализации задать через переменные окружения

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: 9455

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:

1500 транзакций в секунду размером 16КБ; обеспечить высокую доступность (High Availability) данных.

Директория WAL файлов: $HOME/zkw63

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/cje38, $HOME/qdx64
- На основе template0 создать новую базу: leftbrownmom
- 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

Этап 1. Инициализация кластера

```
PGDATA=$HOME/u08/djs10
PGLOCALE=ru_RU.CP1251
PGENCODE=WIN1251
PGUSERNAME=postgres0
PGHOST=pg109
export PGDATA PGLOCALE PGENCODE PGUSERNAME PGHOST

mkdir -p $PGDATA

initdb --locale=$PGLOCALE --encoding=$PGENCODE --
username=$PGUSERNAME
```

```
[postgres0@pg109 ~]$ initdb --locale=$PGLOCALE --encoding=$PGENCODE --username=$PGUSERNAME
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres0".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "ru_RU.CP1251".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres0/u08/djs10... ок
создание подкаталогов... ок
выбирается реализация динамической разделяемой памяти... posix
выбирается значение max_connections по умолчанию... 100
выбирается значение shared_buffers по умолчанию... 128MB
выбирается часовой пояс по умолчанию... Europe/Moscow
создание конфигурационных файлов... ок
выполняется подготовительный скрипт... ок
выполняется заключительная инициализация... ок
сохранение данных на диске... ок

initdb: предупреждение: включение метода аутентификации "trust" для локальных подключений
initdb: подсказка: Другой метод можно выбрать, отредактировав pg_hba.conf или ещё раз запустив initdb с ключом -A, --auth-local или --auth-host.

Готово. Теперь вы можете запустить сервер баз данных:

    pg_ctl -D /var/db/postgres0/u08/djs10 -l файл_журнала start
```

```
pg_ctl -D /var/db/postgres0/u08/djs10 -l logfile start
```

```
[postgres0@pg109 ~]$ pg_ctl -D /var/db/postgres0/u08/djs10 -l logfile start
ожидание запуска сервера.... готово
сервер запущен
```

Этап 2. Конфигурация и запуск сервера БД

Конфигурация pg_hba.conf

1. Способы подключения
   - Unix-domain сокет в режиме peer;
   - сокет TCP/IP, принимать подключения к любому IP-адресу узла
2. Способ аутентификации TCP/IP клиентов: по паролю в открытом виде
3. Остальные способы подключений запретить.

```
# "local" is for Unix domain socket connections only
local   all             all
peer
# IPv4 local connections:
host    all             all             127.0.0.1/32
password
# IPv6 local connections:
host    all             all             ::1/128
password
```

Конфигурация posrgresql.conf

- Номер порта: 9455

```
port = 9455
```

- 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: 1500 транзакций в секунду размером 16КБ; обеспечить высокую доступность (High Availability) данных

- max_connections

```
max_connections = 2000
```

Для поддержки 1500 TPS нам нужно как минимум 1500 открытых соединений. Я взял с запасом в 500 соединений, чтобы был запас для открытых транзакций в пуле соединения

- shared_buffers

```
shared_buffers = 2GB
```

> Если вы используете выделенный сервер с объёмом ОЗУ 1 ГБ и более, разумным начальным значением shared_buffers будет 25% от объёма памяти. Существуют варианты нагрузки, при которых эффективны

> будут и ещё большие значения shared_buffers, но так как PostgreSQL использует и кеш операционной системы, выделять для shared_buffers более 40% ОЗУ вряд ли будет полезно. При увеличении shared_buffers обычно требуется соответственно увеличить max_wal_size, чтобы растянуть процесс записи большого объёма новых или изменённых данных на более продолжительное время.
>
> — Документация

Для значения shared_buffers возьмем 25% от доступной памяти

- temp_buffers

```
temp_buffers = 16MB
```

Нам требуется высокая доступность данных поэтому увеличим значение в 2 раза

- work_mem

Если мы предположим что у нас 1500 TPS размером 16КБ, которые будут созданы в одном соединении то 16КБ * 1500 = 24Мб

округлив получим

```
work_mem = 32MB
```

- checkpoint_timeout

> Уменьшение значений checkpoint_timeout и/или max_wal_size приводит к учащению контрольных точек. Это позволяет ускорить восстановление после краха (поскольку для воспроизведения нужно меньше данных), но с другой стороны нужно учитывать дополнительную нагрузку, возникающую вследствие более частого сброса «грязных» страниц данных на диск
>
> — Документация

Для высокой доступности уменьшим число в 2 раза до 2 минут

```
checkpoint_timeout = 2min
```

- effective_cache_size

> Определяет представление планировщика об эффективном размере дискового кеша, доступном для одного запроса. Это представление влияет на оценку стоимости использования индекса; чем выше это значение, тем больше вероятность, что будет применяться сканирование по индексу, чем ниже, тем более вероятно, что будет выбрано последовательное сканирование.
>
> — Документация

Для высокой доступности надо увеличить кеш чтобы больше операций использовали индекс

```
effective_cache_size = 8GB
```

- fsync

> Если этот параметр установлен, сервер PostgreSQL старается добиться, чтобы изменения были записаны на диск физически, выполняя системные вызовы fsync() или другими подобными методами (см. wal_sync_method). Это даёт гарантию, что кластер баз данных сможет вернуться в согласованное состояние после сбоя оборудования или операционной системы.
>
> — Документация

Для высокой доступности в случае сбоя нам следует оставить значение - true

```
fsync = on
```

- commit_delay

> Параметр commit_delay добавляет паузу (в микросекундах) перед собственно выполнением сохранения WAL. Эта задержка может увеличить быстродействие при фиксировании множества транзакций, позволяя зафиксировать большее число транзакций за одну операцию сохранения WAL, если система нагружена достаточно сильно и за заданное время успевают зафиксироваться другие транзакции.
>
> — Документация

Чтобы увеличить доступность установим значение в 10 мс

```
commit_delay = 10
```

- Директория WAL файлов: $HOME/zkw63

```
archive_mode = on
archive_command = 'cp %p $HOME/zkw63/%f'
```

> Если режим архивации (archive_mode) не был включён при запуске, этот параметр игнорируется. Если значение archive_command — пустая строка (по умолчанию), но archive_mode включён, архивация WAL временно отключается, но сервер продолжает накапливать файлы сегментов WAL в ожидании, что команда будет вскоре определена.
>
> — Документация

- Формат лог-файлов: .csv

```
log_destination = 'csvlog'
```

> В качестве значения log_destination указывается один или несколько методов протоколирования, разделённых запятыми. По умолчанию используется stderr. Если в log_destination включено значение csvlog, то протоколирование ведётся в формате CSV (разделённые запятыми значения).
>
> — Документация

```
logging_collector = on
```

> Параметр включает сборщик сообщений. Это фоновый процесс, который собирает отправленные в stderr сообщения и перенаправляет их в журнальные файлы. Такой подход зачастую более полезен чем запись в syslog, поскольку некоторые сообщения в syslog могут не попасть. (Типичный пример с сообщениями об ошибках динамического связывания, другой пример — ошибки в скриптах типа archive_command.)
>
> — Документация

```
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
```

> При включённом logging_collector задаёт имена журнальных файлов. Значение по умолчанию postgresql-%Y-%m-%d_%H%M%S.log. Если в log_destination включён вывод в формате CSV, то к имени журнального файла будет добавлено расширение .csv. (Если log_filename заканчивается на .log, то это расширение заменится на .csv.)
>
> — Документация

- Уровень сообщений лога: ERROR

```
log_min_messages = error
```

> Управляет минимальным уровнем сообщений, записываемых в журнал сервера. Допустимые значения DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, INFO, NOTICE, WARNING, ERROR, LOG, FATAL и PANIC. Каждый из перечисленных уровней включает все идущие после него. Чем дальше в этом списке уровень сообщения, тем меньше сообщений будет записано в журнал сервера. По умолчанию используется WARNING.
>
> — Документация

- Дополнительно логировать: завершение сессий и продолжительность выполнения команд

```
log_disconnections = on
log_duration = on
```

Включаем протоколирование завершения сеанса и продолжительность каждой завершённой команды.

Этап 3. Дополнительные табличные пространства и наполнение базы

- Создать новые табличные пространства для временных объектов: $HOME/cje38, $HOME/qdx64

```
mkdir -p $HOME/cje38
mkdir -p $HOME/qdx64
```

```
psql -h localhost -p 9455 -U postgres0 -d postgres
```

```
CREATE TABLESPACE cje38 LOCATION '/var/db/postgres0/cje38';
CREATE TABLESPACE qdx64 LOCATION '/var/db/postgres0/qdx64';
```

```
postgres=# CREATE TABLESPACE cje38 LOCATION '/var/db/postgres0/cje38';
CREATE TABLESPACE
postgres=# CREATE TABLESPACE qdx64 LOCATION '/var/db/postgres0/qdx64';
CREATE TABLESPACE
postgres=# \db
            Список табличных пространств
    Имя     | Владелец  |       Расположение
------------+-----------+--------------------------
 cje38      | postgres0 | /var/db/postgres0/cje38
 pg_default | postgres0 |
 pg_global  | postgres0 |
 qdx64      | postgres0 | /var/db/postgres0/qdx64
(4 строки)
```

- На основе template0 создать новую базу: leftbrownmom

```
CREATE DATABASE leftbrownmom WITH TEMPLATE=template0;
```

```
postgres=# CREATE DATABASE leftbrownmom WITH TEMPLATE=template0;
CREATE DATABASE
postgres=# \l
                                                  Список баз данных
    Имя     | Владелец | Кодировка | Провайдер локали |  LC_COLLATE  |  LC_CTYPE   | локаль ICU | Правила ICU |      Права доступа
------------+----------+-----------+------------------+--------------+-------------+------------+-------------+-----------------------
 leftbrownmom | postgres0 | WIN1251   | libc             | ru_RU.CP1251 | ru_RU.CP1251 |            |             |
```

- Создать новую роль, предоставить необходимые права, разрешить подключение к базе.

```
CREATE ROLE new_user WITH LOGIN PASSWORD 'password123!';
GRANT CONNECT ON DATABASE leftbrownmom TO new_user;
```

- От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ т

абличные пространства должны использоваться по назначению. Созданная роль без выданных прав:

```
leftbrownmom=> CREATE TABLE public.test_table1 (
    id SERIAL PRIMARY KEY,
    data TEXT
) TABLESPACE cje38;
ОШИБКА:  нет доступа к схеме public
```

```
psql -h localhost -p 9455 -U postgres0 -d leftbrownmom
```

```
GRANT ALL PRIVILEGES ON SCHEMA public TO new_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO
new_user;
GRANT CREATE ON TABLESPACE cje38, qdx64 TO new_user;
```

```
psql -h localhost -p 9455 -U new_user -d leftbrownmom
```

Срипт наполнения данными:

```
CREATE TABLE test_table1 (
    id SERIAL PRIMARY KEY,
    data TEXT
) TABLESPACE cje38;

CREATE TABLE test_table2 (
    id SERIAL PRIMARY KEY,
    data TEXT
) TABLESPACE qdx64;

INSERT INTO test_table1 (data) VALUES ('Test data 1'), ('Test data
2');
INSERT INTO test_table2 (data) VALUES ('Test data 3'), ('Test data
4');

CREATE TEMP TABLE test_temp_table1 (
                            id SERIAL PRIMARY KEY,
                            data TEXT
) TABLESPACE cje38;

CREATE TEMP TABLE test_temp_table2 (
                            id SERIAL PRIMARY KEY,
                            data TEXT
) TABLESPACE qdx64;

INSERT INTO test_temp_table1 (data) VALUES ('Test data 1'), ('Test
data 2');
INSERT INTO test_temp_table2 (data) VALUES ('Test data 3'), ('Test
data 4');
```

Расположение таблиц по схемам:

```
SELECT
    n.nspname AS schema_name,
    c.relname AS table_name
FROM
    pg_class c
JOIN
    pg_namespace n ON c.relnamespace = n.oid
WHERE
    c.relkind = 'r'
    AND n.nspname != 'pg_catalog'
    AND n.nspname != 'information_schema'
ORDER BY
    schema_name, table_name;
```

```
leftbrownmom=> SELECT
    n.nspname AS schema_name,
    c.relname AS table_name
FROM
    pg_class c
JOIN
    pg_namespace n ON c.relnamespace = n.oid
WHERE
    c.relkind = 'r'
    AND n.nspname != 'pg_catalog'
    AND n.nspname != 'information_schema'
ORDER BY
    schema_name, table_name;
 schema_name |    table_name
-------------+------------------
 pg_temp_3   | test_temp_table1
 pg_temp_3   | test_temp_table2
 public      | test_table1
 public      | test_table2
(4 строки)
```

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

Список табличных простарнств:

```
SELECT spcname AS tablespace_name, pg_tablespace_location(oid) AS
location
FROM pg_tablespace;
```

```
leftbrownmom=> SELECT spcname AS tablespace_name, pg_tablespace_location(oid) AS location
FROM pg_tablespace;
 tablespace_name |       location
-----------------+----------------------
 pg_default      |
 pg_global       |
 cje38           | /var/db/postgres0/cje38
 qdx64           | /var/db/postgres0/qdx64
(4 строки)
```

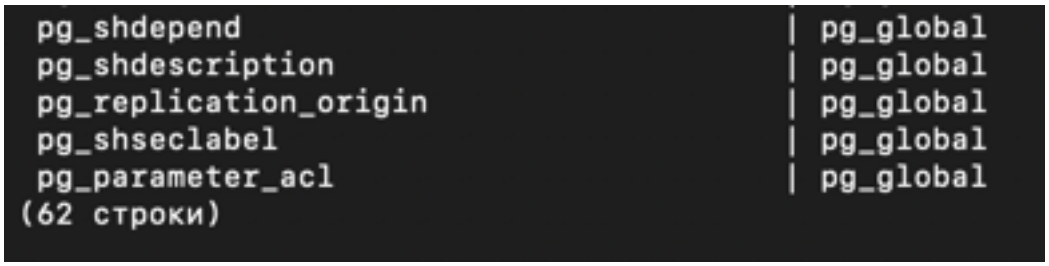
```
SELECT relname AS table_name, spcname AS tablespace_name
FROM pg_class
JOIN pg_tablespace ON pg_class.reltablespace = pg_tablespace.oid
WHERE relkind = 'r';
```

Только таблицы:

```
leftbrownmom=> SELECT relname AS table_name, spcname AS tablespace_name
FROM pg_class
JOIN pg_tablespace ON pg_class.reltablespace = pg_tablespace.oid
WHERE relkind = 'r';
      table_name       | tablespace_name
-----------------------+-----------------
 test_table1           | cje38
 test_table2           | qdx64
 test_temp_table1      | cje38
 test_temp_table2      | qdx64
 pg_authid             | 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      | pg_global
(15 строк)
```

Все объекты:

```
               table_name                | tablespace_name
-----------------------------------------+-----------------
 pg_toast_16394                          | cje38
 pg_toast_16394_index                    | cje38
 test_table1                             | cje38
 pg_toast_16403                          | qdx64
 pg_toast_16403_index                    | qdx64
 test_table2                             | qdx64
 pg_toast_16414                          | cje38
 pg_toast_16414_index                    | cje38
 test_temp_table1                        | cje38
 pg_toast_16423                          | qdx64
 pg_toast_16423_index                    | qdx64
 test_temp_table2                        | qdx64
 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                               | 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
--More--(byte 3252)
```



Вывод

В ходе выполнения лабораторной работы был создан и сконфигурирован кластер БД на выделенном узле, мы познакомились с различными вариантами конфигурации. Также была создана БД, новая роль, табличные пространства и заполнение тестовыми данными.