НАЦИОНАЛЬНО-ИССЛЕДОВАТЕЛЬСКАЯ КОРПОРАЦИЯ ИТМО

Факультет программной инженерии и компьютерной техники

Распределённые системы хранения данных

Лабораторная работа №2

Вариант: 6

Выполнил
студент:

Нестеров Иван Алексеевич

Группа № P33302

Преподаватель:

Шешуков Дмитрий Михайлович

г. Санкт-Петербург

2023

Задание:

На выделенном узле создать и сконфигурировать новый кластер БД, саму БД, табличные пространства и новую роль в соответствии с заданием. Произвести наполнение базы.

Отчёт должен содержать все команды по настройке, а также измененные строки конфигурационных файлов.

Персональный пароль для работы с узлом выдается преподавателем. Обратите внимание, что домашняя директория пользователя /var/postgres/$LOGNAME

Этапы выполнения работы:

Инициализация кластера БД

• Имя узла — pg102.

• Имя пользователя — postgres1.

• Директория кластера БД — $HOME/u01/gsd65.

• Кодировка, локаль — KOI8-R, русская

• Перечисленные параметры задать через аргументы команды.

Конфигурация и запуск сервера БД

• Способ подключения к БД — TCP/IP socket, номер порта 9006.

• Остальные способы подключений запретить.

• Способ аутентификации клиентов — по имени пользователя.

• 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: 1000 транзакций/сек. с записью размером по 8 КБ, акцент на высокую доступность данных;

• Директория WAL файлов — $HOME/u02/gsd65.

• Формат лог-файлов — csv.

• Уровень сообщений лога — ERROR.

• Дополнительно логировать — завершение сессий и продолжительность выполнения команд.

Дополнительные табличные пространства и наполнение

• Создать новые табличные пространства для различных таблиц:

? $HOME/u03/gsd65;

? $HOME/u04/gsd65;

? $HOME/u05/gsd65.

• На основе template0 создать новую базу — greatercapybara.

• От имени новой роли (не администратора) произвести наполнение существующих баз тестовыми наборами данных. Предоставить права по необходимости. Табличные пространства должны использоваться по назначению.

• Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Ход работы:

Опишем логику инициализации кластера в виде shell-скрипта, предварительно создав и заполнив переменные окружения нужными данными, согласно варианту.

```
# initialize necessary system variables

PGDATA=$HOME/u01/gsd65  # directory of database cluster
PGENCODE=KOI8-R         # encoding
PGLOCALE=ru_RU.KOI8-R   # locale
PGUSERNAME=postgres1   # username

export PGDATA PGLOCALE PGENCODE PGSUSERNAME
# export system variables
mkdir -p $PGDATA
# move to directory for cluster storing
chown postgres1 $PGDATA
# change owner of this direcrory
initdb --encoding=$PGENCODE  --locale=$PGLOCALE --username=$PGSUSERNAME
# initialize database cluster
```

Листинг 1. Предварительная настройка терминала для инициализации базы данных.

Отобразим вывод в терминале после запуска команды initdb.

```
[postgres1@pg102 /var]$ initdb --encoding=$PGENCODE  --locale=$PGLOCALE --username=$PGUSERNAME
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres1".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "ru_RU.KOI8-R".
Выбрана конфигурация текстового поиска по умолчанию "russian".
```

```
Контроль целостности страниц данных отключён.

исправление прав для существующего каталога
/var/db/postgres1/u01/gsd65... 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... W-SU
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
Другой метод можно выбрать, отредактировав pg_hba.conf или используя
ключи -A,
--auth-local или --auth-host при следующем выполнении initdb.

Готово. Теперь вы можете запустить сервер баз данных:

    pg_ctl -D /var/db/postgres1/u01/gsd65 -l файл_журнала start

[postgres1@pg102 /var]$
```

Листинг 2. Инициализация базы данных.

Как видим, кластер был успешно создан. Скачаем теперь файлы pg_hba.conf и postgresql.conf и изменим их, чтобы конфигурация кластера указывала указанной во варианте.

Итак, фрагмент pg_hba.conf, изменим данные в конце файла, настроим аутентификацию по имени, такая аутентификация доступна только для TCP/IP подключений, что соответствует нашим требованиям. На первых порах настроим авторизацию по имени пользователя прямиком в pg_hba.conf файле (для root-пользователя, чтобы создать роль и для s312621 пользователя, чтобы выполнить последние задания лабораторной). В дальнейшем же можно будет изменить настройки в pg_ident.conf файле, чтобы подключаться могли все пользователи, имя которых соответствует указанному отображению. В таком случае метод нужно будет заменить на ident. Но пока не будем этого делать.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# connection configuration for lab2:
# ident - auth via username (available only for TCP/IP)
# reject in another cases
host    all             s312621         all                     trust
host    all             postgres1       all                     trust
# "local" is for Unix domain socket connections only
```

```
local   all             all                                     reject
# IPv4 local connections:
host    all             all             127.0.0.1/32            reject
# IPv6 local connections:
host    all             all             ::1/128                 reject
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     reject
host    replication     all             127.0.0.1/32            reject
host    replication     all             ::1/128                 reject
```

Листинг 3. Обновленное содержимое файла pg_hba.conf

Измененные строки файла postgresql.conf приведены в следующем листинге. Поясним мотивацию.

1. Порт для подключения – 9006, согласно варианту.
2. Максимальное количество подключений, равное 100, менять не будем, так как памяти на это достаточно, такое количество также соответствует указанному требованию к высокой доступности данных: есть возможность одновременной работы множества пользователей, причем эта нагрузка на кластер небольшая, поэтому данные высоко доступными для всех пользователей.
3. Размер shared_buffers зададим равным 25% от доступной памяти, то есть 2 ГБ. Выделение большего размера приведет к недостатку памяти для других программ, что приведет к снижению производительности из-за частого swapping-а страниц, что перестанет соответствовать требованию высокой доступности данных.
4. Значение temp_buffers для хранения временных таблиц нельзя делать слишком большим, чтобы избежать неэффективного использования памяти. Вычислить определенное значение сложно, так как специфика использования кластера неизвестна. Выставим 16 МБ в зависимости от конфигурации, чтобы сохранить высокую доступность данных.
5. Аналогично поступим и с work_mem, использующимся для операций чтения и сортировки.
6. Флаг fsync имеет смысл отключать на read-only копиях баз данных, в других случаях нужно включать для повышения отказоустойчивости независимо от конфигурации системы. Поэтому включим его.
7. Изменение задержки перед сохранением WAL имеет смысл только в том случае, если есть возможность протестировать его влияние на общую производительность. В нашем случае оставляем 0 по умолчанию.
8. Значение паузы между точками восстановления по умолчанию не изменяем. Лимит подключений, выставленный ранее не дает оснований снижать ее, а повышение может сильно увеличить время восстановления.
9. Параметр effective_cache_size влияет на эффективность планирования выполнения запросов. Оставляем значение по умолчанию, так как оно не перегружает память и его достаточно для использования индексов. Единственное ограничение – параметр не может быть меньше shared_buffers.

10. Логируем помимо контрольных точек еще и завершение сессий, а также длительность выполнения операций, как указано во варианте.
11. Уровень сообщений логирования выставим error, в соответствии с вариантом.
12. Формат лог-файлов выставим syslog, опять же согласно варианту.

```
port = 9006                             # (change requires restart)
max_connections = 100                   # (change requires restart)
shared_buffers = 2048MB                 # min 128kB
temp_buffers = 16MB                     # min 800kB
work_mem = 16MB                         # min 64kB
dynamic_shared_memory_type = posix      # the default is the first option
fsync = on                              # flush data to disk for crash
safety
commit_delay = 0                        # range 0-100000, in microseconds
checkpoint_timeout = 5min               # range 30s-1d
max_wal_size = 1GB
min_wal_size = 80MB
effective_cache_size = 4GB
log_destination = 'csvlog'
log_min_messages = error                # values in order of decreasing
detail:
log_checkpoints = on
log_disconnections = on
log_duration = on
```

Листинг 4. Измененные строки файла postgresql.conf

Загружаем конфигурационные файлы обратно.

```
[s312621@helios ~/distrubuted-systems-of-data-storage/lab2]$ scp
/home/studs/s312621/distrubuted-systems-of-data-storage/lab2/pg_hba.conf
postgres1@pg102:u01/gsd65/
Password for postgres1@pg102.cs.ifmo.ru:
pg_hba.conf
100% 5084     6.3MB/s   00:00
[s312621@helios ~/distrubuted-systems-of-data-storage/lab2]$ scp
/home/studs/s312621/distrubuted-systems-of-data-
storage/lab2/postgresql.conf postgres1@pg102:u01/gsd65/
Password for postgres1@pg102.cs.ifmo.ru:
postgresql.conf
100%   44KB  23.2MB/s   00:00
[s312621@helios ~/distrubuted-systems-of-data-storage/lab2]$
```

Листинг 5. Загрузка конфигурационных файлов по SCP

Запускаем кластер.

```
[postgres1@pg102 ~]$ pg_ctl start
ожидание запуска сервера....2023-04-06 14:16:26.529 MSK [49986]
СООБЩЕНИЕ:  завершение вывода в stderr
2023-04-06 14:16:26.529 MSK [49986] ПОДСКАЗКА:  В дальнейшем протокол
будет выводиться в "csvlog".
```

```
2023-04-06 14:16:26.529 MSK [49986] СООБЩЕНИЕ:  запускается PostgreSQL
14.2 on amd64-portbld-freebsd13.0, compiled by FreeBSD clang version
11.0.1 (git@github.com:llvm/llvm-project.git llvmorg-11.0.1-0-
g43ff75f2c3fe), 64-bit
2023-04-06 14:16:26.530 MSK [49986] СООБЩЕНИЕ:  для приёма подключений по
адресу IPv6 "::" открыт порт 9006
2023-04-06 14:16:26.530 MSK [49986] СООБЩЕНИЕ:  для приёма подключений по
адресу IPv4 "0.0.0.0" открыт порт 9006
2023-04-06 14:16:26.540 MSK [49986] СООБЩЕНИЕ:  для приёма подключений
открыт Unix-сокет "/tmp/.s.PGSQL.9006"
2023-04-06 14:16:26.561 MSK [49987] СООБЩЕНИЕ:  система БД была
выключена: 2023-04-06 12:48:16 MSK
2023-04-06 14:16:26.590 MSK [49986] СООБЩЕНИЕ:  система БД готова
принимать подключения
 готово
сервер запущен
```

Листинг 6. Запуск кластера после обновления конфигурационных файлов

Авторизовываемся как root-пользователь в консоли psql, как видим, пароль не требуется, консоль открылась, а значит, метод авторизации по имени сработал корректно.

```
[s312621@helios ~]$ psql -h pg102 -d template1 -p 9006 -U postgres1
psql (14.2)
Введите "help", чтобы получить справку.

template1=#
```

Листинг 7. Загрузка конфигурационных файлов по SCP

Создаем необходимые табличные представления по заданным адресам.

```
CREATE TABLESPACE tablespace1 LOCATION '/var/db/postgres1/u03/gsd65';
CREATE TABLESPACE tablespace2 LOCATION '/var/db/postgres1/u04/gsd65';
CREATE TABLESPACE tablespace3 LOCATION '/var/db/postgres1/u05/gsd65';
```

Листинг 8. Создание табличных представлений

Создаем базу данных на основе template0 согласно варианту, а также создаем роль для работы с этой базой данных. Помним, что авторизация для этой роли уже настроена.

```
CREATE DATABASE greatercapybara WITH TEMPLATE = template0;
CREATE ROLE s312621 LOGIN PASSWORD 'PASSWORD';
```

Листинг 9. Создание базы данных и роли для работы в ней

Отключаемся и перезаходим в созданную базу данных от имени нового пользователя.

```
psql -h pg102 -d greatercapybara -p 9006
```

Листинг 10. Авторизация в созданной базе данных по созданной роли

Инициируем создание трех отношений. Убедимся, что составленная выше русская раскладка корректно отображается и здесь.

```
CREATE TABLE квартал
(
    id       SERIAL PRIMARY KEY,
    название TEXT NOT NULL
);

CREATE TABLE квартал_квартал
(
    id          SERIAL PRIMARY KEY,
    квартал1_id INTEGER REFERENCES квартал ON DELETE CASCADE ON UPDATE
CASCADE NOT NULL,
    квартал2_id INTEGER REFERENCES квартал ON DELETE CASCADE ON UPDATE
CASCADE NOT NULL check (квартал2_id != квартал1_id)
);

--Улицы
CREATE TABLE улица
(
    id          SERIAL PRIMARY KEY,
    имя         TEXT
NOT NULL,
    квартал_ID INTEGER REFERENCES квартал ON DELETE CASCADE ON UPDATE
CASCADE NOT NULL
);
```

Листинг 11. Создание тестовых отношений в новой базе данных

Заполняем отношения тестовыми данными.

```
INSERT INTO "квартал" ("id", "название")
VALUES (1, 'Манхеттен'),
       (2, 'Бронкс'),
       (3, 'Куинс'),
       (4, 'Бруклин'),
       (5, 'Стэттен-Айланд'),
       (6, 'Джерси-Сити');
INSERT INTO "квартал_квартал" ("квартал1_id", "квартал2_id")
```

```
VALUES (1, 2),
       (1, 3),
       (1, 4),
       (1, 5),
       (1, 6),
       (2, 3),
       (3, 4),
       (4, 5),
       (5, 6);
INSERT INTO "улица" ("id", "имя", "квартал_id")
VALUES (1, 'Бродвей', 1),
       (2, 'Фалтон Стрит', 1),
       (3, 'Парк Авеню', 1),
       (4, 'Лафаетт Авеню', 2),
       (5, 'Дайер Авеню', 2),
       (6, 'Брон Ривер', 2),
       (7, 'Роквей Авеню', 3),
       (8, 'Куинс Бульвар', 3),
       (9, 'Улица Сергея Довлатова', 3),
       (10, 'Бульвар Роквей', 4),
       (11, 'Белт-Паркуэй', 4),
       (12, 'Променад Ригельмана', 4),
       (13, 'Ошен Террас', 5),
       (14, 'Клов Роуд', 5),
       (15, 'Бойл Плаза', 6),
       (16, 'Палисейд Авеню', 6);
```

Листинг 12. Заполнение отношений тестовыми данными

Выводим содержимое таблиц на экран, чтобы убедиться в отображении русской раскладки и корректной работы базы данных.

```
greatercapybara=> SELECT * FROM квартал;
 id |     название
----+----------------
  1 | Манхеттен
  2 | Бронкс
  3 | Куинс
  4 | Бруклин
  5 | Стэттен-Айланд
  6 | Джерси-Сити
(6 строк)

greatercapybara=> SELECT * FROM квартал_квартал;
 id | квартал1_id | квартал2_id
----+-------------+-------------
  1 |           1 |           2
  2 |           1 |           3
  3 |           1 |           4
  4 |           1 |           5
  5 |           1 |           6
  6 |           2 |           3
```

```
  7 |           3 |           4
  8 |           4 |           5
  9 |           5 |           6
(9 строк)

greatercapybara=> SELECT * FROM улица;
 id |          имя           | квартал_id
----+------------------------+------------
  1 | Бродвей                |          1
  2 | Фалтон Стрит           |          1
  3 | Парк Авеню             |          1
  4 | Лафаетт Авеню          |          2
  5 | Дайер Авеню            |          2
  6 | Брон Ривер             |          2
  7 | Роквей Авеню           |          3
  8 | Куинс Бульвар          |          3
  9 | Улица Сергея Довлатова |          3
 10 | Бульвар Роквей         |          4
 11 | Белт-Паркуэй           |          4
 12 | Променад Ригельмана    |          4
 13 | Ошен Террас            |          5
 14 | Клов Роуд              |          5
 15 | Бойл Плаза             |          6
 16 | Палисейд Авеню         |          6
(16 строк)
```

Листинг 13. Вывод содержимого таблиц на экран

Выведем также список отношений, чтобы убедиться, что владелец схем – указанный выше пользователь с созданной ролью.

```
greatercapybara=> \dt;
                Список отношений
 Схема  |       Имя        |   Тип   | Владелец
--------+------------------+---------+----------
 public | квартал          | таблица | s312621
 public | квартал_квартал  | таблица | s312621
 public | улица            | таблица | s312621
(3 строки)
```

Листинг 14. Вывод списка отношений на экран

Выводим табличные пространства созданного кластера и их содержимое.

```
greatercapybara=> select * from pg_tablespace;
  oid  |   spcname   | spcowner | spcacl | spcoptions
-------+-------------+----------+--------+------------
  1663 | pg_default  |       10 |        |
  1664 | pg_global   |       10 |        |
 16387 | tablespace1 |       10 |        |
```

```
 16388 | tablespace2 |       10 |
 16389 | tablespace3 |       10 |
(5 строк)

greatercapybara=> SELECT c.relname, t.spcname FROM pg_class c JOIN
pg_tablespace t ON c.reltablespace = t.oid;
                 relname                  |  spcname
------------------------------------------+-----------
 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_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_role_member_index        | pg_global
 pg_auth_members_member_role_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_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
 pg_shseclabel                            | pg_global
(43 строки)
```

Листинг 15. Вывод табличных пространств кластера и их содержимого

Завершаем работу на узле, чтобы не потреблять ресурсы зря.

```
[postgres1@pg102 ~/u05]$ pg_ctl stop
2023-04-06 15:55:12.732 MSK [51870] СООБЩЕНИЕ:  получен запрос на быстрое
выключение
ожидание завершения работы сервера....2023-04-06 15:55:12.763 MSK [51870]
СООБЩЕНИЕ:  прерывание всех активных транзакций
2023-04-06 15:55:12.763 MSK [52112] :
2023-04-06 15:55:12.769 MSK [51870] СООБЩЕНИЕ:  фоновый процесс "logical
replication launcher" (PID 51877) завершился с кодом выхода 1
2023-04-06 15:55:12.771 MSK [51872] СООБЩЕНИЕ:  выключение
2023-04-06 15:55:12.876 MSK [51870] СООБЩЕНИЕ:  система БД выключена
 готово
сервер остановлен
[postgres1@pg102 ~/u05]$
```

Листинг 16. Выключение PostgreSQL на узле

Вывод:

В ходе проделанной работы я познакомился с созданием собственного кластера базы данных, настройкой базы данных по указанным явно, а также по очевидным из явных параметрам. Был реализован нетипичный способ авторизации по имени пользователю, что мотивировало значительно лучше понять содержимое файлов конфигурации postrgresql.conf, pg_hba.conf и pg_ident.conf. Основная проблема была в необходимости часто переключаться между окнами терминалов, чтобы неоднократно перезапускать кластер, обновив по SCP его конфигурационные файлы. Практика показала, что настройка собственного кластера – длительный и непростой процесс, требующий времени, терпения и внимательности. В ходе работы также удалось повторить работу с SQL и основательно поработать в shell. Графический интерфейс использовался лишь для изменения содержимого конфигурационных файлов на helios для их последующей загрузки по SCP на узел.