Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина:
«*Распределенные системы хранения данных*»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
Вариант №77807

Выполнил:
Студенты гр. P33111
Емельянов Дмитрий Сергеевич
Герасимов Артём Кириллович

Проверил:
Николаев Владимир Вячеславович

Санкт-Петербург
2024 г.

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/ehz35`
- Кодировка: ISO_8859_5
- Локаль: русская
- Параметры инициализации задать через аргументы команды

Этап 2. Конфигурация и запуск сервера БД

- Способ подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, только localhost
- Номер порта: `9807`
- Остальные способы подключений запретить.
- Способ аутентификации клиентов: по паролю SHA-256
- 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: 500 транзакций в секунду размером 8КБ; обеспечить высокую доступность (High Availability) данных.
- Директория WAL файлов: `$HOME/euu73`
- Формат лог-файлов: `.log`
- Уровень сообщений лога: `INFO`
- Дополнительно логировать: попытки подключения и завершение сессий

Этап 3. Дополнительные табличные пространства и наполнение базы

- Создать новое табличное пространство для индексов: `$HOME/kls16`
- На основе template1 создать новую базу: `lastgoldsoup`
- 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

Номер узла: pg185
Пользователь: postgres0 88f5RQrA

Этап 1

```
# Заходим на сервер
ssh -J s339018@helios.cs.ifmo.ru:2222 postgres0@pg185
# 88f5RQrA

mkdir ehz35
# Этап 1 - инициализация кластера БД
initdb -D $HOME/ehz35 -E ISO_8859_5 --locale=ru_RU.ISO8859-5 -D $HOME/ehz35
```

Тут важно у локали также указать кодировку(*locale -a*), иначе появится такая ошибка:

```
[postgres0@pg185 ~]$ initdb -D $HOME/ehz35 -E ISO_8859_5 --locale=ru_RU  -D $HOME/ehz35
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres0".
От его имени также будет запускаться процесс сервера.

initdb: ошибка: ошибочное имя локали "ru_RU"
```

При успешной инициализации кластера появляется такое сообщение:

```
[postgres0@pg185 ~]$ initdb -D $HOME/ehz35 -E ISO_8859_5 --locale=ru_RU.ISO8859-5 -D $HOME/ehz35
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.ISO8859-5".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres0/ehz35... 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/postgres0/ehz35 -l файл_журнала start
```

Этап 2

pg_hba.conf

Добавляем новую(выделенную строку), где указываем тип пароля. Остальным ставим *reject*

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
# 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
```

postgresql.conf

Меняем порт

```
port = 9807             # (change requires restart)
```

Устанавливаем шифрование паролей (раскоменчиваем строку)

```
password_encryption = scram-sha-256 # scram-sha-256 or md5
```

Директория WAL файлов

Команда локальной оболочки, выполняемая для архивирования завершенного сегмента файла WAL. Если acrhive_mode = off, тогда archive_command - игнорируется. Если archive_command является пустой строкой (по умолчанию) при включенном archive_mode, архивирование WAL временно отключается, но сервер продолжает накапливать файлы сегмента WAL в ожидании, что вскоре будет предоставлена команда.

```
archive_mode = on       # enables archiving; off, on, or always
               # (change requires restart)
archive_command = 'cp %p /var/db/postgres0/euu73/%f'        # command to use to archive a logfile
segment
```

Параметр включает сборщик сообщений (logging collector). 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.)

При включенном logging_collector, определяет каталог, в котором создаются журнальные файлы. Можно задавать как абсолютный путь, так и относительный от каталога данных кластера.

```
# log_destination = 'syslog'
log_destination = 'stderr'      # Valid values are combinations of
                  # stderr, csvlog, syslog, and eventlog,
                  # depending on platform.  csvlog
                  # requires logging_collector to be on.

# This is used when logging to stderr:
logging_collector = on      # Enable capturing of stderr and csvlog
                  # into log files. Required to be on for
                  # csvlogs.
                  # (change requires restart)
```

Уровень сообщений лога

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.

```
log_min_messages = info     # values in order of decreasing detail:
                   #   debug5
                   #   debug4
                   #   debug3
                   #   debug2
                   #   debug1
                   #   info
                   #   notice
                   #   warning
                   #   error
                   #   log
                   #   fatal
                   #   panic
```

Дополнительное логирование: попытки подключения и завершение сессий

```
log_connections = on
log_disconnections = on
```

Максимальное количество соединений *max_connections* не должно превышать количество транзакций(500), чтобы избежать избыточного потребления ресурсов сервера

```
max_connections = 500           # (change requires restart)
```

Кол-во подготовленных транзакций должно быть >= *max_connections*

```
max_prepared_transactions = 500     # zero disables the feature
```

Число временных буферов высчитывается из того расчета, что 8Кб(запись 1 транзакции) * 500 (max транзакций) = 4Мб, но я считаю, что 1 сеанс вряд ли создаст 500 транзакций за 1 секунду, при этом требуется сделать высокопроизводительную систему, поэтому я делаю размер *temp_buffers* и *work_mem* равным 2Мб

```
temp_buffers = 2MB          # min 800kB
work_mem = 2MB              # min 64kB
```

Shared_buffers = 25% * вся память.

На t*emp_buffers* и *work_mem* для 500 транзакций будет тратиться 2Мб*2*500=2Гб, поэтому общая память будет 4Гб и *effective_cache_size=shared_buffers= 1Гб*

```
shared_buffers = 1GB            # min 128kB
effective_cache_size = 1GB
```

При изменении *shared_buffers* следует также изменить *max_wal_size*, чтобы растянуть процесс добавления/изменения большего объема данных

```
max_wal_size = 2GB
```

При увеличении - повышается время восстановления после сбоя, при уменьшении - повышается нагрузка, поэтому оставил дефолтным

```
checkpoint_timeout = 5min       # range 30s-1d
```

Если *off*, то при внезапном отключении питания велик риск потери всех данных, но возрастает производительность, при *on*(дефолтное значение) ниже производительность, но и рисков нет

```
fsync = on              # flush data to disk for crash safety
```

Этап 3

Создание БД

```
CREATE DATABASE lastgoldsoup TEMPLATE template1; # по дефолту и так template1, но мы решили, чтоб наверняка
```

Заходим в новую БД

```
psql -d lastgoldsoup -p 9807
```

Создание табличного пространства

```
CREATE TABLESPACE indexTablespace LOCATION '/var/db/postgres0/kls16';
```

Создаем пользователя

```
CREATE USER student1 WITH PASSWORD '123';
```

Выдача прав новому пользователю

```
GRANT INSERT ON my_table to student1;
GRANT USAGE, SELECT ON SEQUENCE my_table_id_seq to student1;
```

Создание таблицы

```
create table my_table(
   id serial,
   name varchar(10)
);

create index on my_table (name) TABLESPACE indexTablespace;
```

Заходим под новым пользователем

```
psql -d lastgoldsoup -U student1 -h 127.0.0.1 -p 9807
```

Заполнение таблицы

```
insert into my_table(name) values('Dima');
insert into my_table(name) values('Artem');
insert into my_table(name) values('Ivan');
insert into my_table(name) values('Andrey');
```

Список всех табличных пространств кластера и содержащиеся в них объекты

```
SELECT * FROM pg_tablespace;
```

```
  oid  |     spcname      | spcowner | spcacl | spcoptions
-------+------------------+----------+--------+------------
  1663 | pg_default       |       10 |        |
  1664 | pg_global        |       10 |        |
 16389 | indextablespace  |       10 |        |
```

```
SELECT relname FROM pg_class WHERE reltablespace IN (SELECT oid FROM pg_tablespace);
```

```
                relname
---------------------------------------
 pg_toast_1262
 pg_toast_1262_index
 pg_toast_2964
 pg_toast_2964_index
 pg_toast_1213
 pg_toast_1213_index
 pg_toast_1260
 pg_toast_1260_index
 pg_toast_2396
 pg_toast_2396_index
 pg_toast_6000
 pg_toast_6000_index
 pg_toast_3592
 pg_toast_3592_index
 pg_toast_6100
 pg_toast_6100_index
 pg_database_datname_index
 pg_database_oid_index
 pg_db_role_setting_databaseid_rol_index
 pg_tablespace_oid_index
 pg_tablespace_spcname_index
[pg_authid_rolname_index
[pg_authid_oid_index
[pg_auth_members_role_member_index
[pg_auth_members_member_role_index
[pg_shdepend_depender_index
[pg_shdepend_reference_index
[pg_shdescription_o_c_index
[pg_replication_origin_roiident_index
[pg_replication_origin_roname_index
[pg_shseclabel_object_index
[pg_subscription_oid_index
[pg_subscription_subname_index
[pg_authid
[my_table_name_idx
[pg_subscription
[pg_database
[pg_db_role_setting
[pg_tablespace
[pg_auth_members
[pg_shdepend
[pg_shdescription
[pg_replication_origin
[pg_shseclabel
[(44 строки)
```

Вывод

Во время выполнения данной лабораторной работы мы научились создавать и конфигурировать кластер БД PostgreSQL, а также познакомились с созданием и работой табличных пространств и ролей.

Полезные ссылки:
https://postgrespro.ru/docs/postgresql/15/creating-cluster
https://postgrespro.ru/docs/postgresql/16/runtime-config-logging#GUC-LOG-FILENAME
https://postgrespro.ru/docs/postgresql/13/runtime-config-connection
https://postgrespro.ru/docs/postgrespro/16/runtime-config-resource#RUNTIME-CONFIG-RESOURCE-MEMORY

https://habr.com/ru/articles/458952/