МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Лабораторная работа №2

по дисциплине
«Распределённые системы хранения данных»

Вариант - **313318**

Выполнил: студент группы P33302
Сабитов Д.Т.

Преподаватель:
Шешуков Д. М.

Санкт-Петербург
2023 г.

Задание:

Этапы выполнения работы:
Инициализация кластера БД
• Имя узла — pg101.
• Имя пользователя — postgres1.
• Директория кластера БД — $HOME/u01/awk50.
• Кодировка, локаль — ANSI1251, русская
• Перечисленные параметры задать через аргументы команды.
Конфигурация и запуск сервера БД
• Способ подключения к БД — TCP/IP socket, номер порта 9001.
• 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 с аппаратной конфигурацией: оперативная память 2 ГБ, хранение на жёстком диске (HDD);
• Директория WAL файлов — $HOME/u02/awk50.
• Формат лог-файлов — log.
• Уровень сообщений лога — NOTICE.
• Дополнительно логировать — попытки подключения и завершение сессий.
Дополнительные табличные пространства и наполнение
• Создать новые табличные пространства для временных объектов:
? $HOME/u03/awk50;
? $HOME/u04/awk50.
• На основе template1 создать новую базу — lazyblackbox.
• 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.

Выполнение:

Этап 1. Инициализация кластера базы данных:

1) Настройка переменных окружения:
    - PGDATA=$HOME/u01/awk50
    - PGENCODE=WIN1251
    - PGLOCALE=ru_RU.CP1251
    - PGUSERNAME=postgres1
2) Установка значений переменных окружения:
    - export PGDATA PGLOCALE PGENCODE PGUSESNAME
3) Создание директории кластера БД:
    - mkdir -p $PGDATA
4) Изменение системного пользователя:
    - chown postgres1 $PGDATA
5) Инициализация кластера:
    - initdb --encoding=$PGENCODE --locale=$PGLOCALE --username=$PGUSERNAME

```
[postgres1@pg101 ~]$ 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.CP1251".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres1/u01/awk50... 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/awk50 -l файл_журнала start
```

Кластер был успешно создан. Меняем теперь файлы pg_hba.conf и postgresql.conf, чтобы конфигурация кластера была как в варианте.

pg_hba.conf:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

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

postgresql.conf:

Меняем порт для подключения, значение max_connections заданное по умолчанию подходит. Снижать его не за чем, так как памяти хватает.

```
port = 9001                             # (change requires restart)
max_connections = 100                   # (change requires restart)
```

Меняем кодировку паролей на SHA-256.

```
# - Authentication -

#authentication_timeout = 1min          # 1s-600s
password_encryption = scram-sha-256     # scram-sha-256 or md5
#db_user_namespace = off
```

shared_buffers: параметр, который задает количество памяти, которое PostgreSQL будет использовать для кэширования данных из таблиц и индексов в оперативной памяти.

Значение shared_buffers = 25% от доступной памяти. По условию на сервере 2GB RAM. В данном случае 0.5 ГБ. Выделение большего размера приведет к недостатку памяти для других программ, что приведет к снижению производительности из-за частого swapping-а страниц

```
# - Memory -

shared_buffers = 512MB                  # min 128kB
                                        # (change requires restart)
```

temp_buffers: параметр устанавливает максимальный размер оперативной памяти, которую сервер может использовать для хранения временных файлов сессии, создаваемых в процессе выполнения операций сортировки и объединения данных.

Значение temp_buffers для хранения временных таблиц нельзя делать слишком большим, чтобы избежать неэффективного использования памяти. Вычислить определенное значение сложно, так как неизвестно для чего будет использоваться кластер. Выставим в зависимости от конфигурации.

```
temp_buffers = 8MB                      # min 800kB
```

work_mem: параметр отвечает за количество памяти, выделяемой для выполнения операций сортировки, хэширования, агрегирования и других операций обработки данных для каждой сессии. Параметр work_mem для операций чтения и сортировки выставляем так же по умолчанию, исходя из конфигурации

```
work_mem = 4MB                          # min 64kB
```

Изменение задержки перед сохранением WAL имеет смысл только в том случае, если есть возможность протестировать его влияние на общую производительность. Оставляем 0 по умолчанию

```
commit_delay = 0                        # range 0-100000, in microseconds
```

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. Установим его в 1 GB. Это даст базе данных достаточный объем кэша для хранения часто запрашиваемых данных в оперативной памяти, что может повысить производительность запросов и уменьшить нагрузку на дисковую подсистему.

```
effective_cache_size = 1GB
```

fsync: Параметр определяет, включена ли синхронизация записи на диск в PostgreSQL.

Флаг fsync имеет смысл отключать на read-only копиях бд, в других случаях нужно включать для повышения отказоустойчивости независимо от конфигурации системы

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

Параметр checkpoint_timeout в PostgreSQL определяет интервал времени в секундах между запусками процесса контрольной точки (checkpoint). Значение параметра checkpoint_timeout зависит от конкретной нагрузки на базу данных и количества изменений, которые происходят в ней за определенный период времени. Значение по умолчанию не изменяем, т.к. этого значения хвататает чтобы обеспечить сохранность данных при возможных сбоях, но не перегружать систему излишней нагрузкой.

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

Включаем архивирование и указываем директорию, в которую будут копироваться WAL-файлы:

```
# - Archiving -

archive_mode = on             # enables archiving; off, on, or always
                              # (change requires restart)
archive_command = 'cp %p $HOME/u02/awk50/%f'          # command to use to archive a logfile segment
```

Логируем только попытки подключения и завершение сессий

```
#log_checkpoints = off
log_connections = on
log_disconnections = on
#log_duration = off
#log_error_verbosity = default          # terse, default, or verbose messages
#log_hostname = off
#log_line_prefix = '%m [%p] '           # special values:
```

Уровень сообщений логирования

```
# - When to Log -

log_min_messages = notice             # values in order of decreasing detail:
                                      #   debug5
                                      #   debug4
```

Этап 2. Дополнительные табличные пространства и наполнение

Подключаемся к postgresql

```
[postgres1@pg101 ~/u01/awk50]$ pg_ctl -D $HOME/u01/awk50 start
ожидание запуска сервера....2023-04-17 14:17:03.606 MSK [77427] СООБЩЕНИЕ:  завершение вывода в stderr
2023-04-17 14:17:03.606 MSK [77427] ПОДСКАЗКА:  В дальнейшем протокол будет выводиться в "syslog".
 готово
сервер запущен
```

```
[postgres1@pg101 ~/u01/awk50]$ psql -h localhost -p 9001 -U postgres1 postgres
psql (14.2)
Введите "help", чтобы получить справку.

postgres=# _
```

Создаем таблицы и табличные пространства для дальнейшего заполнения.

```
postgres=# create tablespace tempobject1 location '/var/db/postgres1/u03/awk50';
CREATE TABLESPACE
postgres=# create tablespace tempobject2 location '/var/db/postgres1/u04/awk50';
CREATE TABLESPACE
postgres=# \db
            Список табличных пространств
     Имя     | Владелец  |         Расположение
-------------+-----------+------------------------------
 pg_default  | postgres1 |
 pg_global   | postgres1 |
 tempobject1 | postgres1 | /var/db/postgres1/u03/awk50
 tempobject2 | postgres1 | /var/db/postgres1/u04/awk50
(4 строки)

postgres=# _
```

Табличные пространства в PostgreSQL позволяют администраторам баз данных определять местоположения в файловой системе, где могут храниться файлы, представляющие объекты базы данных. Это полезно по крайней мере в двух случаях. Во-первых, если в разделе или томе, на котором был инициализирован кластер, заканчивается пространство и его невозможно расширить, табличное пространство может быть создано в другом разделе. Во-вторых, табличные пространства позволяют администратору использовать знания о схеме использования объектов базы данных для оптимизации производительности

Создаем базу данных

```
postgres=# create database lazyblackbox with template = template1;
CREATE DATABASE
```

Добавляем роль для пользователя danil и выдаем права

```
postgres=# create role danil login password 'password';
CREATE ROLE
```

```
postgres=# grant insert on tableBox,partTableBox1,partTableBox2 to danil;
GRANT
```

```
postgres=# GRANT ALL PRIVILEGES ON TABLESPACE tempobject1,tempobject2 TO danil;
GRANT
```

Переподключаемся к бд от danil

```
[postgres1@pg101 ~/u01/awk50]$ psql -h localhost -p 9001 -U newrole lazyblackbox
psql (14.2)
Введите "help", чтобы получить справку.

lazyblackbox=>
```

```
[postgres1@pg101 ~/u01/awk50]$ psql -h localhost -p 9001 -U danil lazyblackbox
Пароль пользователя danil:
psql (14.2)
Введите "help", чтобы получить справку.

lazyblackbox=>
```

Создаем таблицы

```
lazyblackbox=> CREATE TABLE tableBox (
T,
    created_lazyblackbox(>     id INT,
lazyblackbox(>     created_at DATE NOT NULL
lazyblackbox(> )
PARTITION BY RANGlazyblackbox-> PARTITION BY RANGE ("created_at");
CREATE TABLE
```

```
 lazyblackbox=> CREATE TABLE partTableBox1 PARTITION OF tableBox
     FOR VALUES FROM ('2023-01-01') TO ('2023-03-31')
     TABLESPACE tempobject1;
 CREATE TABLE
 lazyblackbox=> CREATE TABLE partTableBox2 PARTITION OF tableBox
     FOR VALUES FROM ('2023-04-01') TO ('2023-06-30')
     TABLESPACE tempobject2;
 CREATE TABLE
```

Заполняем таблицы и выводим содержимое

```
lazyblackbox=> insert into tableBox(id, created_at) values (1, '2023-01-01'), (2, '2023-04-02');
INSERT 0 2
lazyblackbox=> select * from partTableBox1;
 id | created_at
----+------------
  1 | 2023-01-01
(1 строка)

lazyblackbox=> select * from partTableBox2;
 id | created_at
----+------------
  2 | 2023-04-02
(1 строка)

lazyblackbox=> select * from tableBox;
 id | created_at
----+------------
  1 | 2023-01-01
  2 | 2023-04-02
(2 строки)
```

Выводим табличные пространства:

```
lazyblackbox=> select * from pg_tablespace;
  oid  |    spcname    | spcowner | spcacl | spcoptions
-------+---------------+----------+--------+------------
  1663 | pg_default    |       10 |        |
  1664 | pg_global     |       10 |        |
 16385 | tempobject1   |       10 |        |
 16386 | tempobject2   |       10 |        |
(4 строки)
```

Помимо списка табличных пространств, выведем список содержащихся в них объектов:

```
lazyblackbox=> 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)
```

Завершаем работу на узле, чтобы не потреблять ресурсы зря.

```
[postgres1@pg101 ~/u01/awk50]$ pg_ctl -D $HOME/u01/awk50 stop
ожидание завершения работы сервера.... готово
сервер остановлен
```

Выводы

Во время выполнения лабораторной работы я научилась создавать и конфигурировать кластер БД PostgreSQL. Я познакомилась с созданием и работой табличных пространств и ролей.