Университет ИТМО
Факультет программной инженерии и компьютерной техники

# Распределённые системы хранения данных. Лабораторная работа №2.

Группа: P33121
Студенты: Гиниятуллин Арслан Рафаилович
Бекмухаметов Владислав Робертович
Преподаватель: Афанасьев Дмитрий Борисович
Вариант: 6

2024

## Ключевые слова

База данных, PostgreSQL, кластер, конфигурация.

## Содержание

**1 Цель работы** **1**

**2 Текст задания** **1**
2.1 Описание . . . 1
2.2 Этапы выполнения работы . . . 2

**3 Выполнение этапов работы** **2**
3.1 Инициализация кластера БД . . . 2
3.1.1 Подключение к узлу с helios . . . 2
3.1.2 Создание кластера . . . 3

**4 Конфигурация и запуск сервера БД** **3**
4.1 Настройка способа подключения и аутентификации в файле pg_hba.conf . . . 3
4.2 Настройка параметров сервера в файле postgresql.conf . . . 3
4.2.1 Номер порта для подключения . . . 3
4.2.2 Настройка параметров сервера для OLTP сценария . . . 4
4.2.3 Настройка логов . . . 4

**5 Дополнительные табличные пространства и наполнение** **5**

# 1 Цель работы

Научиться создавать и конфигурировать кластер баз данных PostgreSQL, сами БД, табличные пространства и роли.

# 2 Текст задания

## 2.1 Описание

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.

- Подключение к узлу через helios:

```
ssh -J sXXXXXX@helios.cs.ifmo.ru:2222 postgresY@pgZZZ
```

- С самого helios или из учебных классов:

```
ssh postgresY@pgZZZ
```

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

## 2.2 Этапы выполнения работы

Этапы выполнения работы:

1. Инициализация кластера БД
   - Имя узла — **pg105**
   - Имя пользователя — **postgres0**
   - Директория кластера БД — **$HOME/u01/gsd65**
   - Кодировка, локаль — **KOI8-R, русская**
   - Перечисленные параметры задать через аргументы команды.
2. Конфигурация и запуск сервера БД
   - Способ подключения к БД — локально, номер порта 9006
   - Остальные способы подключений **запретить**
   - Способ аутентификации клиентов — **по имени пользователя**
   - Настроить следующие параметры сервера:

     `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 — **$HOME/u02/gsd65**
   - Формат лог-файлов — **csv**
   - Уровень сообщений лога — **ERROR**
   - Дополнительно логировать — завершение сессий и продолжительность выполнения команд
3. Дополнительные табличные пространства и наполнение
   - Создать новые табличные пространства для различных таблиц:
     - **$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.

# 3 Выполнение этапов работы

## 3.1 Инициализация кластера БД

### 3.1.1 Подключение к узлу с helios

Подключаемся к узлу **pg105** под пользователем **postgres0**

```
ssh postgres0@pg105
```

### 3.1.2 Создание кластера

Создадим соответствующий каталог для будущего кластера PostgreSQL, предоставим права владения пользователю **postgres0**, чтобы была возможность создание подкаталогов, перейдем под его управление.

```
mkdir -p $HOME/u01/gsd65
chown postgres0 $HOME/u01/gsd65
su postgres0
```

Далее инициализируем кластер в ранее созданном каталоге –pgdata=**$HOME/u01/gsd65**, с именем пользователя –username=**postgres0**, кодировкой –encoding=**KOI8R** и локалью –locale=**ru_RU.KOI8-R**.

```
initdb --pgdata=$HOME/u01/gsd65 --username=postgres0 --encoding=KOI8R --locale=ru_RU.KOI8-R
```

# 4 Конфигурация и запуск сервера БД

## 4.1 Настройка способа подключения и аутентификации в файле pg_hba.conf

Для настройки аутентификации клиентов по **локальному Unix-domain сокету**, используя имя пользователя, нужно отредактировать конфигурационный файл **pg_hba.conf**, добавив строку (см. ниже)

```
#TYPE  DATABASE        USER            ADDRESS                 METHOD
 local all             all                                     peer  map=postgres0
```

И в файле **pg_ident.conf** добавим соответствующий маппинг (см. ниже)

```
# MAPNAME       SYSTEM-USERNAME         PG-USERNAME
postgres0       postgres0               postgres0
```

Здесь **local** – указывает на управление подключениями, устанавливаемыми по **Unix-domain сокетам**.

**all** – указывает на то, что запись подходит всем БД и всем пользователям.

**peer** – получает имя пользователя операционной системы клиента из операционной системы и проверяет, соответствует ли оно имени пользователя запрашиваемой базы данных. Доступно только для локальных подключений.

Чтобы исключить типы покдлючений через Unix-domain-сокеты и оставить только локальные TCP/IP подключения, необходимо выставить следующие значения в **postgresql.conf**.

Где *listen_addresses=”* – указывает на TCP/IP адреса, которые должен слушать сервер (в нашем случае таковых нет), исключая Unix-сокеты, а настройка *unix_socket_directories=**’tmp’*** – указывает на каталог Unix-сокета для принятия соединений.

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -
listen_addresses = 'localhost'
unix_socket_directories = '\tmp'
```

## 4.2 Настройка параметров сервера в файле postgresql.conf

### 4.2.1 Номер порта для подключения

Для подключения к БД по порту **9006** добавим в конфигурационный файл **postgresql.conf** следующую строку.

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -
port = 9006
```

### 4.2.2 Настройка параметров сервера для OLTP сценария

**Описание сценария**: **1000 транзакций/сек** с записью размером по **8 КБ**, акцент на высокую доступность данных.

Установим значения **max_connections**=1000 для одновременного обслуживания 1000 подключений. Так как одно соединение може обрабатывать ону транзакцию за промежуток времени.

```
max_connections = 1000
```

Если каждый сеанс создаст по 1 транзакции и чтобы ограничить количество, сколько может выполняться максимально транзакций за раз от всех 1000 сеансов ниже нужно выставить параметр **max_prepared_transactions** на 1000.

```
max_prepared_transactions = 1000
```

Для значения **temp_buffers**, которое задаёт максимальный объём памяти, выделяемой для временных буферов в каждом сеансе, следует выставить в **8KB * 1000 = 8MB**, однако вероятность поддерживания 1000 транзакций в одном сенасе мала, поставим среднее значение в 4MB.

```
temp_buffffers = 4MB
```

Для **work_mem**, соответсвенно, тоже выставим значение в 4MB. Этот параметр Зздаёт базовый максимальный объём памяти, который будет использоваться во внутренних операциях при обработке запросов (например, для сортировки или хеш-таблиц), прежде чем будут задействованы временные файлы на диске.

```
work_mem = 4MB
```

Задаёт объём памяти, который будет использовать сервер баз данных для буферов в разделяемой памяти. 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 использует и кеш операционной системы, выделять для shared_buffers более 40% ОЗУ вряд ли будет полезно.

Теперь имея представление о затратах системы, можно предположить какой объем памяти понадобится системе. 4 GB будет задействовано максимум от work_mem и temp_buffers на 1000 транзакций (то есть 8 GB в сумме). Так как 25% мы будем выделять под sharred_buffers, а оставшуюся память под effective_cache_size. Благоприятным значением системы будет объем памяти 16GB.

25% * 16GB = 4GB - выделяем под shared_buffers. 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. Но при увеличении min_wal_size время восстановление системы будет достаточно высокое.

### 4.2.3 Настройка логов

**Включаем Write ahead log и настраиваем путь для сохранения.**

```
archive_mode = on
archive_command = 'mkdir -p $HOME/study/u02/gsd65 && cp %p $HOME/study/u02/gsd65/%f'
```

**csvlog - формат логов (csv файл). Чтобы csvlog исправно работал нужно включить logging_controller, а для его корректной работы необходимо включить stderr**

```
log_destination = 'csvlog, stderr'
```

**Включаем сборщик сообщений. фоновый процесс, который собирает отправленные в stderr сообщения и перенаправляет их в журнальные файлы**

```
logging_collector = true
log_directory = log
log_min_messages = error
```

**Указываем уровень и дополнительные настройки логов**

```
log_min_messages = error
log_disconnections = on
log_duration = on
```

# 5 Дополнительные табличные пространства и наполнение

Создаем директории для новых таблих пространств. Каталог должен быть пустым и принадлежать пользователю ОС, под которым запущен PostgreSQL.

```
mkdir -p $HOME/u03/gsd65
mkdir -p $HOME/u04/gsd65
mkdir -p $HOME/u05/gsd65
```

Подключаемся к нашему серверу

```
psql -p 9006 -u postgres0 postgres
```

Создаем необходимые табличные пространства. Создавать табличное пространство должен суперпользователь базы данных, но после этого можно разрешить обычным пользователям его использовать. После создания можно проверить успешность командой db

```
CREATE TABLESPACE u03_gsd65 LOCATION '/var/db/postgres0/u03/gsd65';
CREATE TABLESPACE u04_gsd65 LOCATION '/var/db/postgres0/u04/gsd65';
CREATE TABLESPACE u05_gsd65 LOCATION '/var/db/postgres0/u05/gsd65';
\db
```

На основе template0 создаем новую базу данных **greatercapybara**

```
createdb -p 9006 -T template1 greatercapybara
```

Создаем тестовые таблицы. Для указания табличного пространства можно использовать TABLESPACE, а можно использовать параметр default_tablespace

```
CREATE TABLE xxx ( id int primary key, name text ) TABLESPACE u03_gsd65;
CREATE TABLE yyy ( id int primary key, name text, xxx_id int references xxx(id)) TABLESPACE u04_
CREATE TABLE zzz ( id int primary key, name text, yyy_id int references yyy(id)) TABLESPACE u05_
```

Создаем тестового пользователя и выдаем необходимые права

```
CREATE USER s000000 WITH PASSWORD '1111';
GRANT CONNECT ON DATABASE greatercapybara TO s000000;
GRANT CREATETABLE ON SCHEMA public TO s000000;

GRANT INSERT on xxx,yyy,zzz TO s000000;
```

Наполнение данными

```
psql -U s000000 -d greatercapybara

INSERT INTO xxx (id, name)
VALUES (1, 'name1'), (2, 'name2'), (3, 'name3');

INSERT INTO yyy (id, name, xxx_id)
VALUES (1, 'name4', 1), (2, 'name5', 2);

INSERT INTO zzz (id, name, yyy_id)
VALUES (1, 'name6', 1);
```

Список всех табличных пространств кластера и содержащиеся в них объекты

```
SELECT ts.spcname AS tablespace_name, NULL AS table_name, NULL AS object_type
FROM pg_tablespace ts
UNION ALL
SELECT ts.spcname AS tablespace_name, c.relname AS table_name, 'table' AS object_type
FROM pg_tablespace ts
        JOIN pg_class c ON ts.oid = c.reltablespace
WHERE c.relkind = 'r'
UNION ALL
SELECT ts.spcname AS tablespace_name, c.relname AS index_name, 'index' AS object_type
FROM pg_tablespace ts
        JOIN pg_class c ON ts.oid = c.reltablespace
        JOIN pg_index i ON c.oid = i.indexrelid
UNION ALL
SELECT ts.spcname AS tablespace_name, c.relname AS sequence_name, 'sequence' AS object_type
FROM pg_tablespace ts
        JOIN pg_class c ON ts.oid = c.reltablespace
WHERE c.relkind = 'S'
ORDER BY tablespace_name, table_name;
```