Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина: Администрирование систем управления базами данных

Лабораторная работа №2

Вариант 77812

Выполнил:
Серебренникова В. В.

Группа:
P33202

Проверил:
Николаев В. В.

Санкт-Петербург
2024

Оглавление

Описание задания

Вариант 77812
Номер узла: pg189, пользователь: postgres0, пароль выдан преподавателем на личную почту.

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/dyu42
- Кодировка: UTF8
- Локаль: английская

Параметры инициализации задать через переменные окружения

Этап 2. Конфигурация и запуск сервера БД

1. Способы подключения:
    a. 1) Unix-domain сокет в режиме peer;
    b. 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла
2. Номер порта: 9812
3. Способ аутентификации TCP/IP клиентов: по паролю в открытом виде
4. Остальные способы подключений запретить.
5. Настроить следующие параметры сервера БД:
    a. max_connections
    b. shared_buffers
    c. temp_buffers
    d. work_mem
    e. checkpoint_timeout
    f. effective_cache_size
    g. fsync
    h. commit_delay

6. Параметры должны быть подобраны в соответствии со сценарием OLTP:
    a. 500 транзакций в секунду размером 8КБ;
    b. обеспечить высокую доступность (High Availability) данных.
7. Директория WAL файлов: $PGDATA/pg_wal
8. Формат лог-файлов: .log
9. Уровень сообщений лога: INFO
10. Дополнительно логировать: попытки подключения и завершение сессий

Этап 3. Дополнительные табличные пространства и наполнение базы

1. Создать новые табличные пространства для временных объектов:
    a. $HOME/sui26
    b. $HOME/ipk91
2. На основе template0 создать новую базу: nicepinklake
3. Создать новую роль, предоставить необходимые права, разрешить подключение к базе.
4. От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ табличные пространства должны использоваться по назначению.
5. Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Отчет

Подготовка

Подключимся к узлу, выделенному нам преподавателем:

```
C:\Users\admin>ssh -J s338828@helios.cs.ifmo.ru:2222 postgres0@pg189
Password:
The authenticity of host 'pg189 (<no hostip for proxy command>)' can't be established.

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'pg189' (ECDSA) to the list of known hosts.
Password for postgres0@pg189.cs.ifmo.ru:
Last login: Wed Apr 24 12:16:53 2024 from 192.168.10.80
[postgres0@pg189 ~]$
```

Рис. 1 - Подключение к узлу

Этап 1

Необходимо инициализировать кластер баз данных (далее - кластер БД) со следущими параметрами:

- Директория кластера: $HOME/dyu42
- Кодировка: UTF8
- Локаль: английская

Для того, чтобы инициализировать кластер в указанной директории, необходимо ее создать. Воспользуемся командой mkdir –p $HOME/dyu42. Флаг -p расшифровывается как parents и нужен для того, чтобы все родительские директории, указанные в команде, при необходимости тоже создались.

```
[postgres0@pg189 ~]$ mkdir -p $HOME/dyu42
[postgres0@pg189 ~]$ ls
070424          backups         cleanup.sh      dyu42           postgres0.dump
```

Рис. 2 - Создание директории

Делаем пользователя postgres0 владельцем директории командой chown postgres0 $HOME/dyu42. После этого переходим к инициализации кластера БД. Нам четко заданы директория и кодировка, а локаль необходимо выбрать из существующих с английским языком. Список локалей можно вывести командой locale –a.

```
[postgres0@pg189 ~]$ chown postgres0 $HOME/dyu42
[postgres0@pg189 ~]$ locale -a
C
C.UTF-8
POSIX
af_ZA.ISO8859-1
af_ZA.ISO8859-15
af_ZA.UTF-8
am_ET.UTF-8
ar_AE.UTF-8
ar_EG.UTF-8
ar_JO.UTF-8
ar_MA.UTF-8
ar_QA.UTF-8
```

Рис. 3 - Выбор локали

Английских локалей много (Австралия, Америка, Великобритания и т. д.), и я взяла на себя ответственность и выбрала британский английский.

Инициализируем кластер БД командой initdb --encoding=UTF8 --locale=en_GB.UTF-8 --username=postgres0 -D $HOME/dyu42. Флаг -D нужен для задания директории кластера.

```
[postgres0@pg189 ~]$ initdb --encoding=UTF8 --locale=en_GB.UTF-8 --username=postgres0 -D $HOME/dyu42
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres0".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "en_GB.UTF-8".
Выбрана конфигурация текстового поиска по умолчанию "english".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres0/dyu42... 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/dyu42 -l файл_журнала start
```

Рис. 4 - Инициализация кластера баз данных

Этап 2

На втором этапе необходимо следующим образом сконфигурировать сервер баз данных:

- Способы подключения:
  - 1) Unix-domain сокет в режиме peer;
  - 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла

- Способ аутентификации TCP/IP клиентов: по паролю в открытом виде
- Остальные способы подключений запретить.

Настроим способы подключения.

Способы подключения должны быть указаны в конфигурационном файле pg_hba.conf.

- 1) Unix-domain сокет в режиме peer;
- 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла

```
[postgres0@pg189 ~]$ cd dyu42
[postgres0@pg189 ~/dyu42]$ ls
base                pg_logical          pg_stat             pg_wal
global              pg_multixact        pg_stat_tmp         pg_xact
pg_commit_ts        pg_notify           pg_subtrans         postgresql.auto.conf
pg_dynshmem         pg_replslot         pg_tblspc           postgresql.conf
pg_hba.conf         pg_serial           pg_twophase
pg_ident.conf       pg_snapshots        PG_VERSION
```

Рис. 5 - Содержимое директории dyu42

Если мы прочитаем файл pg_hba.conf командой cat pg_hba.conf, то увидим следующее содержимое:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     trust
host    replication     all             127.0.0.1/32            trust
host    replication     all             ::1/128                 trust
[postgres0@pg189 ~/dyu42]$
```

Рис. 6 - Состав автоматически созданного файла pg_hba.conf

Заменим содержимое файла при помощи редактора vi в соответствии с вариантом:

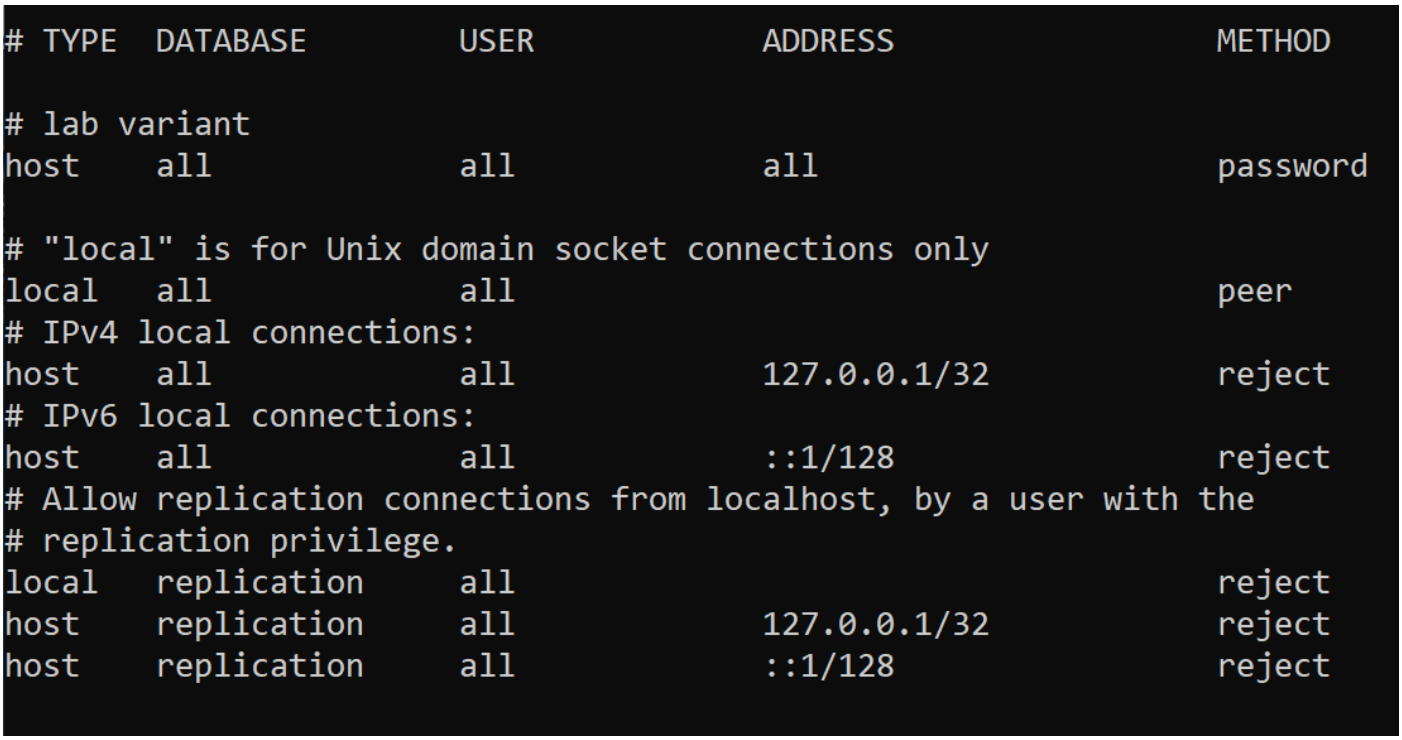

Рис. 7 - Новый состав файла pg_hba.conf

Теперь зададим номер порта 9812 и настроим 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.

Сценарий OLTP (On-Line Transaction Proccessing) направлен на быструю обработку большого количества небольших транзакций.

Порт по умолчанию: 5432. Меняем порт на 9812.

Параметр **max_connections** по умолчанию равен 100. Скорее всего, учитывая, что операций будет много, пользователей, проводящих операции, тоже будет много. Предположим, что 500 транзацкий в секунду совершают 250 пользователей (по полсекунды на транзакцию). Повысила значение max_connections до 250.

Рис. 8 - Изменение параметров ”порт“, ”максимальное кол-во соединений”

Параметр **shared_buffers** по умолчанию равен 128 МБ. Параметр задает количество памяти, которое будет использоваться для кэширования данных из таблиц и индексов в оперативной памяти от всего выделенного ОЗУ. В документации написано “Если вы используете выделенный сервер с объёмом ОЗУ 1 ГБ и более, разумным начальным значением shared_buffers будет 25% от объёма памяти”. temp_buffers, work_mem оба равны 4 Мб, и на 500 транзакций каждый будет задействовать около 2 Гб. Поделим ОЗУ между temp_buffers, work_mem, shared_buffers и effective_cache_size, (где shared_buffers получает 25% ОЗУ) тогда получается, что общий объем ОЗУ = 8 Гб, а объем shared_buffers = 2 Гб.

Рис. 9 - Изменение параметра ”разделяемые буферы”

Параметр **temp_buffers** по умолчанию равен 8MB. Нам не требуется большой объем временных буферов, так как в OLTP системе операции небольшие, и мы можем оставить значение по умолчанию или уменьшить - я уменьшила до 4 МБ.

Рис. 10 - Изменение параметра ”временные буферы”

Параметр **work_mem** по умолчанию равен 4 МБ. Так как OLTP работает с большим количеством небольших операций, я оставила work_mem неизменной.

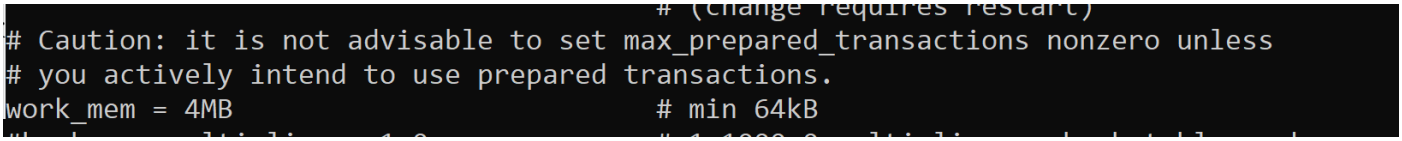

Рис. 11 - Изменение параметра ”рабочая память”

Параметр **checkpoint_timeout** по умолчанию равен 5 минут. Операций много и проходят они часто, поэтому checkpoint_timeout я понизила до 1 минуты.

Рис. 12 - Изменение параметра ” checkpoint_timeout”

Параметр **effective_cache_size** по умолчанию равен 4 GB. Параметр определяет представление планировщика о размере дискового кэша, досупного для одного запроса. Этот параметр оценочный и не задаёт размер резервируемого в ядре дискового кеша. Я поставила параметр равным 2 GB.

```
effective_cache_size = 2GB
```

Рис. 12 - Изменение параметра ”эффективный размер кэша”

Параметр **fsync** по умолчанию со значением on. Параметр определяет, происходит ли синхронизация записи на диск. Не меняла данный параметр, так как нам нужно сохранять многочисленные изменения.

```
fsync = on                              # flush data to disk for crash safety
                                        # (turning this off can cause
                                        # unrecoverable data corruption)
```

Рис. 14 - Изменение параметра ”fsync”

Параметр **commit_delay** по умолчанию равен 0. Этот параметр добавляет паузу перед сохранением WAL. Я не стала менять параметр commit_delay, так как нам не нужно ждать перед записью во Write-Ahead Buffers.

```
commit_delay = 0                        # range 0-100000, in microseconds
#commit_siblings = 5                    # range 1-1000
```

Рис. 15 - Изменение параметра ”commit delay”

В следующем пункте отчета зададим следующие параметры сервера:

- Директория WAL файлов: $PGDATA/pg_wal
- Формат лог-файлов: .log
- Уровень сообщений лога: INFO
- Дополнительно логировать: попытки подключения и завершение сессий

Директория pg_wal является директорией по умолчанию для хранения WAL-файлов. Укажем его в Archiving:

```
archive_mode = on               # enables archiving; off, on, or always
                                # (change requires restart)
archive_command = 'cp %p $HOME/dyu42/pg_wal'            # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
```

Рис. 16 - Задание директории хранения WAL-файлов

Зададим формат log-файлов .log:

```
# - Where to Log -

log_destination = 'stderr'
#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)
```

Рис. 17 - Изменение формата log-файлов

Изменим уровень сообщений лога на INFO:

```
# - When to Log -

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

Рис. 18 - Изменение уровня сообщений лога

Поставим on на логировании подключения и завершения сессий:

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

Рис. 19 - Включение дополнительных параметров

Этап 3

Требуется создать новые табличные пространства для следующих временных объектов:

- $HOME/sui26
- $HOME/ipk91

Создание табличных пространств выполняется при подключенной базе данных. Для этого сначала запустим сервер:

```
[postgres0@pg189 ~]$ pg_ctl -D $HOME/dyu42 -l logfile start
ожидание запуска сервера.... готово
сервер запущен
[postgres0@pg189 ~]$
```

Рис. 20 - Запуск сервера

Теперь выполним подключение к базе данных:

```
[postgres0@pg189 ~/dyu42]$ pg_ctl -D $HOME/dyu42 -l logfile start
ожидание запуска сервера.... готово
сервер запущен
[postgres0@pg189 ~/dyu42]$ psql -h localhost -p 9812 -U postgres0 postgres
Пароль пользователя postgres0:
psql (14.2)
Введите "help", чтобы получить справку.

postgres=# ^Z
```

Рис. 21 - Подключение к базе данных

Создадим директории для табличных пространств sui26, ipk91:

```
[postgres0@pg189 ~/dyu42]$ mkdir -p $HOME/sui26
[postgres0@pg189 ~/dyu42]$ mkdir -p $HOME/ipk91
```

Рис. 22 - Создание директорий для табличных пространств

Создадим табличные пространства sui26, ipk91:

```
postgres=# create tablespace sui26 location '/var/db/postgres0/sui26';
CREATE TABLESPACE
postgres=# create tablespace ipk91 location '/var/db/postgres0/ipk91';
CREATE TABLESPACE
postgres=#
```

Рис. 23 - Создание табличных пространств

Создадим базу данных “nicepinklake”:

```
postgres=# create database nicepinklake template template0;
CREATE DATABASE
postgres=#
```

Рис. 24 - Создание базы данных

Создадим пользователя nicepinkuser и дадим ему права на подключение к базе данных и на пользование табличными пространствами:

```
postgres=# create role nicepinkuser with login password 'pink';
CREATE ROLE
postgres=# grant connect on database nicepinklake to nicepinkuser;
GRANT
```

Рис. 25 - Создание пользователя

```
postgres=# grant all on tablespace sui26 to nicepinkuser;
GRANT
postgres=# grant all on tablespace ipk91 to nicepinkuser;
GRANT
postgres=#
```

Рис. 26 - Передача привилегий

Войдем в базу данных под пользователем nicepinkuser:

```
[postgres0@pg189 ~/dyu42]$ psql -h localhost -p 9812 -U nicepinkuser nicepinklake
Пароль пользователя nicepinkuser:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=>
```

sshРис. 27 - Подключение под новым пользователем

Создадим скрипт для создания таблиц в базе:

```
[postgres0@pg189 ~/dyu42]$ vi createscript
create table colors (
id integer primary key,
color text
);

create table lakes (
id integer primary key,
name text,
color_id integer references references colors(id)
);

create table people (
id integer primary key,
name text,
age integer,
occupation text
);
```

Рис. 28 - Скрипт для создания таблиц в базе

Создадим скрипт для заполнения таблиц базы:

```
insert into colors (id, color)
values
('pink'),
('green'),
('blue');

insert into lakes (name, color_id)
values
('pinklake', 1),
('greenlake', 2),
('bluelake', 3);

insert into people (name, age, occupation)
values
('john', 40, 'doctor'),
('mark', 35, 'florist'),
('michelle', 36, 'librarian');
```

Рис. 29 - Скрипт для заполнения базы

Создадим таблицы:

```
nicepinklake=> \i createscript
CREATE TABLE
CREATE TABLE
CREATE TABLE
```

Рис. 30 - Создание таблиц

Заполним таблицы:

```
nicepinklake=> \i insertscript
INSERT 0 3
INSERT 0 3
INSERT 0 3
```

Рис. 31 - Заполнение таблиц

Создадим временный объект (таблицу) в табличном пространстве sui26. Таблица во втором пространстве создается аналогично, обе таблицы заполняются данными.

```
create temporary table student (
id serial,
name text
) tablespace sui26;
```

Рис. 33 - Создание временного объекта
в табличном пространстве sui26

Проверим, что все работает:

```
nicepinklake=> \i temptable1
CREATE TABLE
nicepinklake=> \i temptable2
CREATE TABLE
nicepinklake=> \i tempinsert1
INSERT 0 3
nicepinklake=> \i tempinsert2
INSERT 0 3
nicepinklake=>
```

Рис. 34 - Заполнение таблиц

Выведем информацию о существующих табличных пространствах из таблицы pg_tablespace:

```
  oid  |  spcname   | spcowner |                      spcacl                      | spcoptions
-------+------------+----------+--------------------------------------------------+------------
  1663 | pg_default |       10 |                                                  |
  1664 | pg_global  |       10 |                                                  |
 16384 | sui26      |       10 | {postgres0=C/postgres0,nicepinkuser=C/postgres0} |
 16385 | ipk91      |       10 | {postgres0=C/postgres0,nicepinkuser=C/postgres0} |
(4 строки)
```

Рис. 35 - Табличные пространства

Выведем информацию обо всех объектах, содержащихся в табличных пространствах:

```
nicepinklake=> select c.relname, t.spcname from pg_class c join pg_tablespace t on c.reltablespace = t.oid;
                relname                  |  spcname
-----------------------------------------+-----------
 pg_toast_16526                          | sui26
 pg_toast_16526_index                    | sui26
 student                                 | sui26
 pg_toast_16533                          | ipk91
 pg_toast_16533_index                    | ipk91
 exam                                    | ipk91
 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
 pg_toast_16509                          | sui26
 pg_toast_16509_index                    | sui26
 student                                 | sui26
 pg_toast_16516                          | ipk91
 pg_toast_16516_index                    | ipk91
 exam                                    | ipk91
(55 строк)

nicepinklake=>
```

Рис. 36 - Объекты, содержащиеся в табличных пространствах

Изменения

Сделаем так, чтобы объекты выводились не таблицей, а строками (рис. 37). Теперь в нашем выводе присутствует так же табличное пространство pg_default. Следующие объекты выводятся при вводе следующего запроса:

```
nicepinklake=# select t.spcname, coalesce(string_agg(c.relname, ',')) from pg_tablespace t
nicepinklake-# left join pg_class c on c.reltablespace = t.oid group by t.spcname order by t.spcname;
  spcname   |

                                                coalesce
------------+-----------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
 ipk91      |
 pg_default |
 pg_global  | 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_toas
t_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_shdepen
d_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,pg_subscription,pg_database,pg_db_rol
e_setting,pg_tablespace,pg_auth_members,pg_shdepend,pg_shdescription,pg_replication_origin,pg_shseclabel
 sui26      |
(4 строки)
```

Рис. 37 - Объекты, содержащиеся в табличных пространствах, через запятую

Попробуем подключиться к базе данных nicepinklake через пользователя nicepinkuser по TCP/IP, выйдя из узла. Запускаем сервев, выходим из узла командой exit, заходим на helios.se.ifmo.ru (ssh s338828@helios.se.ifmo.ru -p 2222) и заходим под пользователем nicepinkuser. На этот раз мы указываем в параметре -h (host) не localhost, а наш узел (pg189). Команда выглядит следующим образом: psql –h pg189 –p 9812 –U nicepinkuser nicepinklake. Видно, что подключение прошло успешно:

```
[s338828@helios ~]$ psql -h pg189  -p 9812 -U nicepinkuser nicepinklake
Пароль пользователя nicepinkuser:
psql (14.7, сервер 14.2)
Введите "help", чтобы получить справку.

nicepinklake=>
```

Рис. 37 - Успешное подключение по TCP/IP

Так как мы завершили работу, вернемся в узел и остановим работу сервера:

```
[postgres0@pg189 ~]$ pg_ctl -D $HOME/dyu42 stop
ожидание завершения работы сервера.... готово
сервер остановлен
[postgres0@pg189 ~]$
выход
Connection to pg189 closed.
```

Рис. 38 - Завершение работы сервера

Выводы

В ходе лабораторной работы я научилась создавать и настраивать кластеры баз данных и сами базы данных, подключаться к серверам, изменять уровни доступа к базам данных и конфигурировать кластеры баз данных через конфигурационные файлы. Я провела работу по созданию скриптов, табличных пространств, заполнению таблиц и расширила свои знаняи в области систем управления базами данных.