Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Отчёт

по лабораторной работе №4

по дисциплине «Распределенные системы хранения данных»

вариант 48752

Выполнили:
Нестеренко К. М., группа P3316
Хоробрых Д. Е., группа P3316

Преподаватель: Николаев В. В.

Санкт-Петербург

~ 2025 ~

Задание: Цель работы - ознакомиться с методами и средствами построения отказоустойчивых решений на базе СУБД Postgres; получить практические навыки восстановления работы системы после отказа.
Работа рассчитана на двух человек и выполняется в три этапа: настройка, симуляция и обработка сбоя, восстановление.

Требования к выполнению работы

- В качестве хостов использовать одинаковые виртуальные машины.
- В первую очередь необходимо обеспечить сетевую связность между ВМ.
- Для подключения к СУБД (например, через psql) использовать отдельную виртуальную или физическую машину.
- Демонстрировать наполнение базы и доступ на запись на примере **не менее, чем двух** таблиц, столбцов, строк, транзакций и клиентских сессий.

Этап 1. Конфигурация

Развернуть postgres на двух узлах в режиме трансляции логов. Не использовать дополнительные пакеты. Продемонстрировать доступ в режиме чтение/запись на основном сервере. Продемонстрировать, что новые данные синхронизируются на резервный сервер.

Этап 2. Симуляция и обработка сбоя

2.1 Подготовка:

- Установить несколько клиентских подключений к СУБД.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

2.2 Сбой:

Симулировать неожиданное отключение основного узла - выполнить Power Off виртуальной машины.

2.3 Обработка:

- Найти и продемонстрировать в логах релевантные сообщения об ошибках.
- Выполнить переключение (failover) на резервный сервер.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 3. Восстановление

- Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.
- Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.
- Восстановить исправную работу узлов в исходной конфигурации (в соответствии с этапом 1).
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 1. Конфигурация

На master:

```
sudo -u vboxuser /usr/lib/postgresql/17/bin/initdb -D ~/Desktop/pgdata

chmod -R 750 ~/Desktop/pgdata

echo "listen_addresses = '*'

wal_level = replica

archive_mode = on

archive_command = 'scp %p vboxuser@192.168.56.102:/home/vboxuser/Desktop/wals/%f.tmp && ssh vboxuser@192.168.56.102 mv /home/vboxuser/Desktop/wals/%f.tmp /home/vboxuser/Desktop/wals/%f'" > ~/Desktop/pgdata/postgresql.conf

sudo -u vboxuser /usr/lib/postgresql/17/bin/pg_ctl -D ~/Desktop/pgdata start

pg_basebackup -D ~/Desktop/pgdata -Fp -Xs -P

scp -r ~/Desktop/backup/ vboxuser@192.168.56.102:~/Desktop/backup
```

На standby:

```
cp -r ~/Desktop/backup/ ~/Desktop/pgdata/

chmod -R 750 ~/Desktop/pgdata

echo "listen_addresses = '*'

hot_standby = on

restore_command = 'echo RESTORE: %f >> /tmp/restore.log && cp /home/vboxuser/Desktop/wals/%f %p'" > ~/Desktop/pgdata/postgresql.conf

touch ~/Desktop/pgdata/standby.signal

sudo -u vboxuser /usr/lib/postgresql/17/bin/pg_ctl -D ~/Desktop/pgdata start
```

```
postgres=# select * from test_data;
 id |                name
----+-----------------------------------
  1 | a
  2 | b
  3 | c
  4 | test
  5 | pleaaaaaseeeee
  6 | 100
  7 | nu vot seichas
  8 | sdlkfmasl;kmfaslkmfasl;kfmsad;lfkm
  9 | <3
 10 | its the last
 11 | really?????
(11 ёЄЁюъ)


postgres=# insert into test_data (name) values ('check');
INSERT 0 1
postgres=# select pg_switch_wal();
 pg_switch_wal
---------------
 0/22000420
(1 ёЄЁюър)


postgres=#
postgres=# select pg_switch_wal();
 pg_switch_wal
---------------
 0/23000000
(1 ёЄЁюър)
```

```
postgres=# select * from test_data;
 id |                name
----+-----------------------------------
  1 | a
  2 | b
  3 | c
  4 | test
  5 | pleaaaaaseeeee
  6 | 100
  7 | nu vot seichas
  8 | sdlkfmasl;kmfaslkmfasl;kfmsad;lfkm
  9 | <3
 10 | its the last
 11 | really?????
(11 ёЄЁюъ)


postgres=# select * from test_data;
 id |                name
----+-----------------------------------
  1 | a
  2 | b
  3 | c
  4 | test
  5 | pleaaaaaseeeee
  6 | 100
  7 | nu vot seichas
  8 | sdlkfmasl;kmfaslkmfasl;kfmsad;lfkm
  9 | <3
 10 | its the last
 11 | really?????
 12 | check
(12 ёЄЁюъ)
```

Этап 2. Симуляция и обработка сбоя

Покажем, что репликация работает успешно. С двух клиентов запишем разные данные в таблицы, удостоверимся, что они появились на реплике

```
CREATE TABLE test1(
   id SERIAL PRIMARY KEY,
   data TEXT
);
CREATE TABLE test2(
   id SERIAL PRIMARY KEY,
   value INT
);
BEGIN;
INSERT INTO test1(data) VALUES('first data');
INSERT INTO test2(value) VALUES(10);
COMMIT;
BEGIN;
INSERT INTO test1(data) VALUES('second data');
```

```
INSERT INTO test2(value) VALUES(20);

COMMIT;

SELECT * FROM test1;

SELECT * FROM test2;
```

Выполним power off master.

```
postgres=# INSERT INTO test1(data) VALUES('second data');
FATAL:  terminating connection due to administrator command
сервер неожиданно закрыл соединение
        Скорее всего сервер прекратил работу из-за сбоя
        до или в процессе выполнения запроса.
Подключение к серверу потеряно. Попытка восстановления неудачна.
Подключение к серверу потеряно. Попытка восстановления неудачна.
```

Подключимся к реплике и выполним

```
sudo -u vboxuser /usr/lib/postgresql/17/bin/pg_ctl promote -D ~/Desktop/pgdata/
```

Проверим, что экземпляр не находится в режиме recovery

```
select pg_is_in_recovery();
```

```
vboxuser@rshd-4--standby:~$ sudo -u vboxuser /usr/lib/postgresql/17/bin/pg_ctl promote -D ~/
Desktop/pgdata/
waiting for server to promote.... done
server promoted
vboxuser@rshd-4--standby:~$ psql -d postgres
psql (17.4 (Ubuntu 17.4-1))
Type "help" for help.

postgres=# select pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 f
(1 row)

postgres=#
```

Подключимся к реплике и попробуем вставить данные в базу через нее

```
PS C:\Users\horob> & 'C:\Program Files\PostgreSQL\15\bin\psql' -h localhost -p 25432 -d post
gres -U vboxuser
psql (15.3, сервер 17.4 (Ubuntu 17.4-1))
ПРЕДУПРЕЖДЕНИЕ: psql имеет базовую версию 15, а сервер - 17.
                Часть функций psql может не работать.
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
                страницы Windows (1251).
                8-битовые (русские) символы могут отображаться некорректно.
                Подробнее об этом смотрите документацию psql, раздел
                "Notes for Windows users".
Введите "help", чтобы получить справку.

postgres=# insert into test1 (data) values ('third data');
INSERT 0 1
postgres=# select * from test1;
 id |    data
----+------------
  1 | first data
  2 | second data
 34 | third data
(3 ёЄЁюъы)


postgres=#
```

Этап 3. Восстановление

Сделаем backup с реплики, восстановим по ней master

```
pg_basebackup -D /tmp/replica_backup -Fp -Xs -P
```

Запустим master, остановим standby, сделаем backup уже с мастера

```
xenon@main:~$ systemctl stop postgresql
xenon@main:~$ sudo systemctl stop postgresql
[sudo] пароль для xenon:
xenon@main:~$ sudo mv /var/lib/postgresql/17/main /var/lib/postgresql/17/main_ol
d
xenon@main:~$ sudo mkdir -p /tmp/main_restore
xenon@main:~$ sudo chown -p /tmp/main_restore
chown: неверный ключ — «p»
По команде «chown --help» можно получить дополнительную информацию.
xenon@main:~$ sudo chown xenon:xenon /tmp/main_restore
xenon@main:~$ hostname -I
10.0.2.15 192.168.100.3 fd17:625c:f037:2:dee4:756:7e61:d612 fd17:625c:f037:2:a00
:27ff:feac:3b3a
xenon@main:~$ ls /tmp/main_restore
xenon@main:~$ ls /tmp/main_restore
replica_backup
xenon@main:~$ sudo cp -r /tmp/main_restore/replica_backup /var/lib/postgresql/17
/main
[sudo] пароль для xenon:
xenon@main:~$ sudo chown -R postgres:postgres /var/lib/postgresql/17/main
xenon@main:~$ sudo systemctl start postgresql
xenon@main:~$ psql -h localhost -U postgres -d postgres
psql (17.4 (Ubuntu 17.4-1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression:
off, ALPN: postgresql)
Type "help" for help.

postgres=# \dt
          List of relations
```

Перебросим backup на standby, и аналогично первому этапу настроим реплику на режим трансляции логов.

Вывод: в ходе выполнения лабораторной работы была настроена отказоустойчивая система на базе PostgreSQL с использованием трансляции логов между двумя виртуальными

машинами. Проверена работоспособность основного и резервного узлов, подтверждена синхронизация данных. Получены практические навыки настройки репликации, обработки сбоев и восстановления работоспособности кластера.