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

Отчёт
Лабораторная №3
по дисциплине «Распределенные системы хранения данных»
Вариант 43987

Выполнила: Нестеренко К. М., группа P3316

Преподаватель: Николаев В.В.

Санкт-Петербург
2025

Текст задания

Цель работы - настроить процедуру периодического резервного копирования базы данных, сконфигурированной в ходе выполнения лабораторной работы №2, а также разработать и отладить сценарии восстановления в случае сбоев.

Узел из предыдущей лабораторной работы используется в качестве основного. Новый узел используется в качестве резервного. Учётные данные для подключения к новому узлу выдаёт преподаватель. В сценариях восстановления необходимо использовать копию данных, полученную на первом этапе данной лабораторной работы.

Требования к отчёту

Отчет должен быть самостоятельным документом (без ссылок на внешние ресурсы), содержать всю последовательность команд и исходный код скриптов по каждому пункту задания. Для демонстрации результатов приводить команду вместе с выводом (самой наглядной частью вывода, при необходимости).

Этап 1. Резервное копирование

- Настроить резервное копирование с основного узла на резервный следующим образом:

  Периодические полные копии + непрерывное архивирование.

  Включить для СУБД режим архивирования WAL; настроить копирование WAL (scp) на резервный узел; настроить полное резервное копирование (pg_basebackup) по расписанию (cron) раз в неделю. Созданные полные копии должны сразу копироваться (scp) на резервный хост. Срок хранения копий на основной системе - 1 неделя, на резервной - 4 недели. По истечении срока хранения, старые архивы и неактуальные WAL должны автоматически уничтожаться.

- Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:
  - Средний объем новых данных в БД за сутки: `1000МБ`.
  - Средний объем измененных данных за сутки: `250МБ`.
- Проанализировать результаты.

Этап 2. Потеря основного узла

Этот сценарий подразумевает полную недоступность основного узла. Необходимо восстановить работу СУБД на РЕЗЕРВНОМ узле, продемонстрировать успешный запуск СУБД и доступность данных.

Этап 3. Повреждение файлов БД

Этот сценарий подразумевает потерю данных (например, в результате сбоя диска или файловой системы) при сохранении доступности основного узла. Необходимо выполнить полное восстановление данных из резервной копии и перезапустить СУБД на ОСНОВНОМ узле.

Ход работы:

- Симулировать сбой:
  - удалить с диска директорию WAL со всем содержимым.
- Проверить работу СУБД, доступность данных, перезапустить СУБД, проанализировать результаты.
- Выполнить восстановление данных из резервной копии, учитывая следующее условие:
  - исходное расположение директории PGDATA недоступно - разместить данные в другой директории и скорректировать конфигурацию.
- Запустить СУБД, проверить работу и доступность данных, проанализировать результаты.

Этап 4. Логическое повреждение данных

Этот сценарий подразумевает частичную потерю данных (в результате нежелательной или ошибочной операции) при сохранении доступности основного узла. Необходимо выполнить восстановление данных на ОСНОВНОМ узле следующим способом:

- Генерация файла на резервном узле с помощью pg_dump и последующее применение файла на основном узле.

Ход работы:

- В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
- Зафиксировать время и симулировать ошибку:
  - удалить любые две таблицы (DROP TABLE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

Выполнение

Этап 1. Резервное копирование

- Настроить резервное копирование с основного узла на резервный следующим образом:

  Периодические полные копии + непрерывное архивирование.

  Включить для СУБД режим архивирования WAL; настроить копирование WAL (scp) на резервный узел; настроить полное резервное копирование (pg_basebackup) по расписанию (cron) раз в неделю. Созданные полные копии должны сразу копироваться (scp) на резервный хост. Срок хранения копий на основной системе - 1 неделя, на резервной - 4 недели. По истечении срока хранения, старые архивы и неактуальные WAL должны автоматически уничтожаться.

Включение архивирования WAL:

```
# - Archiving -
archive_mode = on             # enables archiving; off, on, or always
                              # (change requires restart)
#archive_library = ''         # library to use to archive a WAL file
                              # (empty string indicates archive_command should
                              # be used)
archive_command = 'scp %p postgres0@pg110:/var/lib/pgsql/wal_archive/%f'          # c
```

```
#------------------------------------------------------------------------------
# WRITE-AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = replica                     # minimal, replica, or logical
                                        # (change requires restart)
```

После внесения изменений выполнена перезагрузка PostgreSQL:

```
pg_ctl restart
```

Создание каталогов для резервных копий и WAL:

На резервном сервере:

```
mkdir -p /var/db/postgres0/wal_archive
chmod 700 /var/db/postgres0/wal_archive
```

На основном сервере:

```
mkdir -p /var/db/postgres0/backups
```

Создадим скрипт `pg_backup.sh` для выполнения полного резервного копирования и архивации WAL:

```
#!/bin/bash

BACKUP_DIR="/var/db/postgres0/backups"
WAL_ARCHIVE_DIR="/var/db/postgres0/wal_archive"
REMOTE_SERVER="postgres0@pg110:/var/db/postgres0/backups"

mkdir -p $BACKUP_DIR
mkdir -p $WAL_ARCHIVE_DIR

BACKUP_DIR_NAME="backup_$(date +%F)"
```

```
echo "Создание резервной копии..."
pg_basebackup -D "$BACKUP_DIR/$BACKUP_DIR_NAME" -Ft -z -P -X stream -U
postgres0 -h 192.168.11.103 -p 9581

echo "Копирование резервной копии на резервный сервер..."
scp -r "$BACKUP_DIR/$BACKUP_DIR_NAME" $REMOTE_SERVER

echo "Удаление старых резервных копий на основной системе..."
find $BACKUP_DIR -type d -mtime +7 -exec rm -rf {} \;

echo "Удаление старых резервных копий на резервной системе..."
ssh postgres0@pg110 "find /var/db/postgres0/backups -type d -mtime +28 -exec
rm -rf {} \;"

echo "Архивирование и копирование WAL на резервный сервер..."
cp /var/db/postgres0/pg_wal/* $WAL_ARCHIVE_DIR/
scp -r $WAL_ARCHIVE_DIR/* $REMOTE_SERVER/wal_archive/

echo "Удаление старых WAL архивов..."
find $WAL_ARCHIVE_DIR -type f -mtime +7 -exec rm -f {} \;
ssh postgres0@pg110 "find /var/db/postgres0/wal_archive -type f -mtime +28 -
exec rm -f {} \;"

echo "Резервное копирование завершено"
```

Добавим задачу в `crontab` для автоматического выполнения резервного копирования:

```
crontab -e
```

И настроим выполнение скрипта каждый понедельник в 2:00:

```
0 2 * * 1 /var/db/postgres0/pg_backup.sh
```

```
crontab: installing new crontab
[postgres0@pg103 ~]$ crontab -l
0 2 * * 1 /var/db/postgres0/pg_backup.sh
```

Расчеты:

- Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:
  - Средний объем новых данных в БД за сутки: `1000МБ`.
  - Средний объем измененных данных за сутки: `250МБ`.
- Проанализировать результаты.

День 1: размер бд = 1000 МБ. Каждый день база растёт на 1000 МБ новых + 250 МБ изменённых → увеличение = 1250 МБ в день, всего 30 дней.

$$S_{30} = \frac{30}{2} \cdot (2 \cdot 1000 + (30 - 1) \cdot 1250) = 573750 \text{ МБ} = 560{,}3 \text{ ГБ}$$

Это достаточно высокая нагрузка на хранилище.

Этап 2. Потеря основного узла

Этот сценарий подразумевает полную недоступность основного узла. Необходимо восстановить работу СУБД на РЕЗЕРВНОМ узле, продемонстрировать успешный запуск СУБД и доступность данных.

На резервном сервере были созданы необходимые каталоги для хранения резервных копий и WAL-архивов:

```
mkdir -p ~/jtp68
mkdir -p ~/wal_archive
mkdir -p ~/backups
```

Архивы WAL хранились в каталоге `tua27` на основном сервере, копируем их в каталог `~/wal_archive/` на резервный сервер:

```
scp -r ~/tua27/* postgres0@pg110:~/wal_archive/
```

Копируем резервную копию с основного узла на резервный сервер:

```
scp -r ~/backups/backup_2025-04-06 postgres0@pg110:~/backups/
```

Распаковка резервной копии в каталог данных `~/jtp68`:

```
tar -xzf base.tar.gz -C ~/jtp68
```

После распаковки данных был создан файл `recovery.signal` для активирования процесса восстановления:

```
touch ~/jtp68/recovery.signal
```

В конфигурационный файл `postgresql.conf` добавим параметр `restore_command`, чтобы PostgreSQL мог восстановить WAL-файлы:

```
echo "restore_command = 'cp ~/wal_archive/%f %p'" >> ~/jtp68/postgresql.conf
```

И запустим кластер:

```
pg_ctl -D ~/jtp68 start
```

```
[postgres0@pg110 ~/backups/backup_2025-04-06]$ pg_ctl -D ~/jtp68 start
ожидание запуска сервера....2025-04-06 10:50:23.138 MSK [17657] СООБЩЕНИЕ:  передача вывода в протокол процессу сбора протоколов
2025-04-06 10:50:23.138 MSK [17657] ПОДСКАЗКА:  В дальнейшем протоколы будут выводиться в каталог "log".
 готово
сервер запущен
```

Проверим подключение к бд и наличие данных:

```
psql -h pg103 -p 9581 -d darkbrowncity -U new_role
```

```
[postgres0@pg110 ~/backups/backup_2025-04-06]$ psql -h pg103 -p 9581 -d darkbrowncity -U new_role
Пароль пользователя new_role:
psql (16.4)
Введите "help", чтобы получить справку.

darkbrowncity=> \dt
              Список отношений
 Схема  |     Имя     |   Тип   | Владелец
--------+-------------+---------+-----------
 public | test_table1 | таблица | postgres0
 public | test_table2 | таблица | postgres0
(2 строки)
```

Этап 3. Повреждение файлов БД

Этот сценарий подразумевает потерю данных (например, в результате сбоя диска или файловой системы) при сохранении доступности основного узла. Необходимо выполнить полное восстановление данных из резервной копии и перезапустить СУБД на ОСНОВНОМ узле.

Ход работы:

- Симулировать сбой:
  - удалить с диска директорию WAL со всем содержимым.
- Проверить работу СУБД, доступность данных, перезапустить СУБД, проанализировать результаты.
- Выполнить восстановление данных из резервной копии, учитывая следующее условие:
  - исходное расположение директории PGDATA недоступно - разместить данные в другой директории и скорректировать конфигурацию.
- Запустить СУБД, проверить работу и доступность данных, проанализировать результаты.

Для симуляции сбоя было удалено содержимое директории WAL. Конкретно была удалена директория `tua27`, которая использовалась для хранения WAL файлов.

```
rm -rf /var/db/postgres0/jtp68/tua27
```

После удаления директории WAL, система не смогла работать корректно, так как требуется восстановление WAL файлов:

```
pg_ctl -D /var/db/postgres0/jtp68 start
```

```
ВАЖНО,XX000,"требуемый каталог WAL ""pg_wal"" не существует",,,,,,,,,"
СООБЩЕНИЕ,00000,"стартовый процесс (PID 20413) завершился с кодом выхо,
```

Результат: СУБД не запустилась из-за отсутствия WAL файлов, и возникла ошибка.

Поскольку исходное расположение данных стало недоступным, создадим новую директорию для размещения восстановленных данных:

```
mkdir /var/db/postgres0/new_pgdata
```

Восстановим данные из каталога резервных копий в новую директорию:

```
rsync -avv ~/backups/backup_2025-04-06/ /var/db/postgres0/new_pgdata/
```

В файле `postgresql.conf` изменим параметр `data_directory`, чтобы указать путь к новой директории:

```
data_directory = '/var/db/postgres0/new_pgdata'
```

Создадим файл `recovery.signal` для активирования процесса восстановления:

```
touch ~/new_pgdata/recovery.signal
```

Также был добавлен параметр `restore_command`:

```
echo "restore_command = 'cp /var/db/postgres0/new_pgdata/pg_wal/%f %p'" >
/var/db/postgres0/new_pgdata/postgresql.conf
```

И запустим новый кластер:

```
pg_ctl -D /var/db/postgres0/new_pgdata start
```

```
[postgres0@pg103 ~]$ pg_ctl -D /var/db/postgres0/new_pgdata status
pg_ctl: сервер работает (PID: 62687)
/usr/local/bin/postgres "-D" "/var/db/postgres0/new_pgdata"
[postgres0@pg103 ~]$
```

Результат: Сервер успешно запустился.

Проверим бд:

```
darkbrowncity=> \dt
              Список отношений
 Схема  |     Имя     |   Тип   | Владелец
--------+-------------+---------+----------
 public | test_table1 | таблица | postgres0
 public | test_table2 | таблица | postgres0
(2 строки)
```

Этап 4. Логическое повреждение данных

Этот сценарий подразумевает частичную потерю данных (в результате нежелательной или ошибочной операции) при сохранении доступности основного узла. Необходимо выполнить восстановление данных на ОСНОВНОМ узле следующим способом:

- Генерация файла на резервном узле с помощью pg_dump и последующее применение файла на основном узле.

Ход работы:

- В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
- Зафиксировать время и симулировать ошибку:
  - удалить любые две таблицы (DROP TABLE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

Исходные данные в таблицах:

```
darkbrowncity=# select * from test_table1;
 id | name | value
----+------+-------
  1 | A    |     1
  2 | B    |     2
  3 | C    |     3
(3 строки)

darkbrowncity=# select * from test_table2;
 id | name | value
----+------+-------
  1 | A    |     1
  2 | B    |     2
  3 | C    |     3
(3 строки)
```

Добавляем новые данные:

```
darkbrowncity=# INSERT INTO test_table1 (name, value) VALUES
  ('AATESTEAA', 266661),
  ('BBTESTEBB', 22);

INSERT INTO test_table2 (name, value) VALUES
  ('AATESTEAA', 26662),
  ('BBTESTEBB', 23);
INSERT 0 2
INSERT 0 2
```

Генерируем дамп на резервном узле:

```
pg_dump -h pg103 -p 9581 -U new_role -d darkbrowncity -Fc -f
/var/db/postgres0/save_dump/darkbrowncity_$(date +"%Y-%m-%d_%H-%M-%S").dump
```

И удаляем таблицы на основном:

```
darkbrowncity=# DROP TABLE IF EXISTS test_table1;
DROP TABLE
darkbrowncity=# DROP TABLE IF EXISTS test_table2;
DROP TABLE
```

```
darkbrowncity=# \dt
Отношения не найдены.
```

Копируем дамп на основной узел:

```
rsync -avz postgres0@pg110:/var/db/postgres0/save_dump/darkbrowncity_2025-04-21_04-57-20.dump /var/db/postgres0/save_dump/
```

Восстанавливаем данные из дампа:

```
pg_restore -h /tmp -p 9581 -U postgres0 -d darkbrowncity -v /var/db/postgres0/save_dump/darkbrowncity_2025-04-21_04-57-20.dump
```

```
[postgres0@pg103 ~]$ pg_restore -h /tmp -p 9581 -U postgres0 -d darkbrowncity -v /var/db/postgres0/save_dump/darkbrowncity_2025-04-21_04-57-20.dump
pg_restore: подключение к базе данных для восстановления
pg_restore: создаётся TABLE "public.test_table1"
pg_restore: создаётся SEQUENCE "public.test_table1_id_seq"
pg_restore: создаётся SEQUENCE OWNED BY "public.test_table1_id_seq"
pg_restore: создаётся TABLE "public.test_table2"
pg_restore: создаётся SEQUENCE "public.test_table2_id_seq"
pg_restore: создаётся SEQUENCE OWNED BY "public.test_table2_id_seq"
pg_restore: создаётся DEFAULT "public.test_table1 id"
pg_restore: создаётся DEFAULT "public.test_table2 id"
pg_restore: обрабатываются данные таблицы "public.test_table1"
pg_restore: обрабатываются данные таблицы "public.test_table2"
pg_restore: выполняется SEQUENCE SET test_table1_id_seq
pg_restore: выполняется SEQUENCE SET test_table2_id_seq
pg_restore: создаётся CONSTRAINT "public.test_table1 test_table1_pkey"
pg_restore: создаётся CONSTRAINT "public.test_table2 test_table2_pkey"
pg_restore: создаётся ACL "public.TABLE test_table1"
pg_restore: создаётся ACL "public.SEQUENCE test_table1_id_seq"
pg_restore: создаётся ACL "public.TABLE test_table2"
pg_restore: создаётся ACL "public.SEQUENCE test_table2_id_seq"
```

В результате получаем восстановленные данные:

```
darkbrowncity=# \dt
              Список отношений
 Схема  |     Имя     |   Тип   | Владелец
--------+-------------+---------+-----------
 public | test_table1 | таблица | postgres0
 public | test_table2 | таблица | postgres0
(2 строки)

darkbrowncity=# select * from test_table1;
 id |   name    | value
----+-----------+--------
  1 | A         |      1
  2 | B         |      2
  3 | C         |      3
  4 | AATESTEAA | 266661
  5 | BBTESTEBB |     22
(5 строк)

darkbrowncity=# select * from test_table2;
 id |   name    | value
----+-----------+-------
  1 | A         |     1
  2 | B         |     2
  3 | C         |     3
  4 | AATESTEAA | 26662
  5 | BBTESTEBB |    23
(5 строк)
```

Выводы

В ходе лабораторной была выполнена настройка периодического резервного копирования базы данных, сконфигурированной в ходе выполнения лабораторной работы №2, а также разработаны сценарии восстановления в случае сбоев.