Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 3 по РСХД

Вариант 51486

Группа: P3316
Выполнили:
Сиразетдинов, Шпинева
Проверил:
Николаев В.В.

г. Санкт-Петербург
2025

Задание

Цель работы

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.

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 () 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: 350МБ. Средний объем измененных данных за сутки: 800МБ.

- 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 дополнительных табличных пространств недоступно - разместить 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:
  - в любой таблице с внешними ключами подменить значения ключей на случайные (INSERT, UPDATE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

Выполнение

Этап 1. Резервное копирование

На основном узле зададим параметры для архивации WAL файлов и их копирования на резервный узел:

```
wal_level = replica
archive_mode = on
archive_command = 'scp %p postgres1@pg114:/var/db/postgres1/
wal_archive/%f'
```

Подключимся как суперпользователь

```
psql -h localhost -p 9455 -U postgres0 -d postgres
```

Создадим роль replicator с правами на репликацию:

```
CREATE ROLE replicator WITH REPLICATION LOGIN PASSWORD
'pass123';
```

Напишем скрипт для создания копии:

```
#!/bin/bash
# Директория для хранения резервных копий на основном узле
LOCAL_BACKUP_DIR="/var/db/postgres0/backups/$(date +%Y-%m-%d_%H-%M-
%S)"
# Директория для хранения резервных копий на резервном узле
REMOTE_BACKUP_DIR="/var/db/postgres1/backups/$(date +%Y-%m-%d_%H-
%M-%S)"
# Создание директории на основном узле
mkdir -p $LOCAL_BACKUP_DIR
# Создание директории на резервном узле
ssh pg1 "mkdir -p $REMOTE_BACKUP_DIR"

# Выполнение резервного копирования на основном узле
pg_basebackup -D $LOCAL_BACKUP_DIR -Ft -z -Xs -P -U replicator -h
pg109 -p 9455
# Проверка успешности выполнения
if [ $? -eq 0 ]; then
  echo "Резервное копирование успешно завершено на основном узле:
$LOCAL_BACKUP_DIR"
  # Копирование резервной копии на резервный узел
  scp -r $LOCAL_BACKUP_DIR/* pg1:$REMOTE_BACKUP_DIR/
  # Проверка успешности копирования
  if [ $? -eq 0 ]; then
    echo "Резервная копия успешно перенесена на резервный узел:
```

```
$REMOTE_BACKUP_DIR"
    # Удаление старых резервных копий на основном узле (старше 1
недели)
    find /var/db/postgres0/backups -type d -mtime +7 -exec rm -rf
{} \;
    echo "Старые резервные копии удалены на основном узле"
    # Удаление старых резервных копий на резервном узле (старше 4
недель)
    ssh pg1 "find /var/db/postgres1/backups -type d -mtime +28 -
exec rm -rf {} \;"
    echo "Старые резервные копии удалены на резервном узле"
  else
    echo "Ошибка при переносе резервной копии на резервный узел"
    exit 1
  fi
else
  echo "Ошибка при выполнении резервного копирования на основном
узле"
  exit 1
fi
```

Добавим права на выполнение

```
chmod +x /var/db/postgres0/script1.sh
```

Настроим выполнение скрипта с помощью кроны раз в неделю

```
crontab -e
0 3 * * 1 /var/db/postgres0/script1.sh
```

```
crontab: installing new crontab
```

Подсчет объемов и анализ результатов

Условия:

- Средний объем новых данных в БД за сутки: 350МБ.
- Средний объем измененных данных за сутки: 800МБ.

Расчет размеров бекапов

В сутки будет 350 МБайт новых данных

Воспользуемся арифметической прогрессией

$$S_{\text{back}} = \frac{2 * 0 + 350 * (30 - 1)}{2} * 30 = 152250 \text{ МБ} = 149 \text{ ГБ}$$

Размер бекапов за месяц - 149 Гб

Расчет размеров wal архивов

wal_segsize = 16 Мб

Объем изменений - 800 Мб -> 50 сегментов = размер wal_archive за сутки - 800Мб

$$S_{\text{wal}} = \frac{(30-1)*800}{2} * 30 = 348000 \text{ Мб} = 340 \text{ Гб}$$

Итоговый размер

$$S = S_{\text{backup}} + S_{\text{wal}} = 340 + 149 = 489 \text{ Гб}$$

Итого через месяц получаем 489 Гб бекапов

Этап 2. Потеря основного узла

Для восстановления воссоздадим файловую структуру кластера и табличного пространства на резервном узле из нашего бэкапа:

```
mkdir -p /var/db/postgres1/u08/djs10
cp -r ~/backups/2025-04-06_20-13-07/* /var/db/postgres1/u08/djs10
chmod 750 /var/db/postgres1/u08/djs10
# Распакуем резервную копию:
cd ~/u08/djs10
tar -xzf base.tar.gz
tar -xzf pg_wal.tar.gz
tar -xzf 16387.tar.gz
tar -xzf 16388.tar.gz
# Укажем в postgresql.conf команду для загрузки wal-файлов:
echo "restore_command = 'cp /var/db/postgres1/u08/djs10/%f %p'"
>> /var/db/postgres1/u08/djs10/postgresql.conf
# Создадим в директории кластера файл, сигнализирующий о
восстановлении:
touch ~/u08/djs10/recovery.signal
# Запустим резервный кластер:
pg_ctl -D ~/u08/djs10 start
# Проверим статус:
pg_ctl -D ~/u08/djs10 status
```

```
[postgres1@pg114 ~/u08/djs10]$ ls
16387.tar.gz    16388.tar.gz    backup_manifest base.tar.gz     pg_wal.tar.gz
[postgres1@pg114 ~/u08/djs10]$ tar -xzf base.tar.gz
tar -xzf pg_wal.tar.gz
tar -xzf 16387.tar.gz
tar -xzf 16388.tar.gz
# Укажем в postgresql.conf команду для загрузки wal-файлов:
echo "restore_command = 'cp /var/db/postgres1/u08/djs10/%f %p'" >> /var/db/postgres1/u08/djs10/postg
resql.conf
# Создадим в директории кластера файл, сигнализирующий о восстановлении:
touch ~/u08/djs10/recovery.signal
# Запустим резервный кластер:
pg_ctl -D ~/u08/djs10 start
# Проверим статус:
pg_ctl -D ~/u08/djs10 status
ожидание запуска сервера....2025-04-06 20:20:26.699 MSK [68331] СООБЩЕНИЕ:  передача вывода в проток
ол процессу сбора протоколов
2025-04-06 20:20:26.699 MSK [68331] ПОДСКАЗКА:  В дальнейшем протоколы будут выводиться в каталог "l
og".
... готово
сервер запущен
pg_ctl: сервер работает (PID: 68331)
/usr/local/bin/postgres "-D" "/var/db/postgres1/u08/djs10"
[postgres1@pg114 ~/u08/djs10]$
```

Проверим таблицы в восстановленной бд:

```
psql -h pg109 -p 9455 -U new_user -d leftbrownmom
\d
```

```
[postgres1@pg114 ~/u08/djs10]$ psql -p 9455 -U new_user -d leftbrownmom -h pg114
Пароль пользователя new_user:
psql (16.4)
Введите "help", чтобы получить справку.

leftbrownmom=> \d
                  Список отношений
 Схема  |        Имя         |        Тип         | Владелец
--------+--------------------+--------------------+----------
 public | test_table1        | таблица            | new_user
 public | test_table1_id_seq | последовательность | new_user
 public | test_table2        | таблица            | new_user
 public | test_table2_id_seq | последовательность | new_user
(4 строки)
```

Этап 3. Повреждение файлов БД

Удалим директорию с WAL файлами и запустим сервер

```
ервано; последний момент работы: 2025-04-06 21:08:18 MSK",,,,,,,,,,"","startup",,0
2025-04-06 21:08:27.763 MSK,,,76274,,67f2c31b.129f2,2,,2025-04-06 21:08:27 MSK,,0,ВАЖНО,XX000,"требуемый каталог WAL ""pg_wal"" не существует",,,,,,,,,"","startup",,0
2025-04-06 21:08:27.765 MSK,,,76270,,67f2c31b.129ee,6,,2025-04-06 21:08:27 MSK,,0,СООБЩЕНИЕ,00000,"стартовый процесс (PID 76274) завершился с кодом выхода 1",,,,,,,,,"","postmaster",,0
2025-04-06 21:08:27.765 MSK,,,76270,,67f2c31b.129ee,7,,2025-04-06 21:08:27 MSK,,0,СООБЩЕНИЕ,00000,"прерывание запуска из-за ошибки в стартовом процессе",,,,,,,,,"","postmaster",,0
2025-04-06 21:08:27.766 MSK,,,76270,,67f2c31b.129ee,8,,2025-04-06 21:08:27 MSK,,0,СООБЩЕНИЕ,00000,"система БД выключена",,,,,,,,,"","postmaster",,0
```

Восстановим данные в новую директорию

```
mkdir -p ~/u08_new/djs10
chmod 750 ~/u08_new/djs10
cd ~/u08_new/djs10
# Распакуем резервную копию:
```

```
tar -xzf ~/backups/2025-04-06_19-29-44/base.tar.gz -C ~/u08_new/
djs10
mkdir -p ~/u08_new/djs10/pg_wal
tar -xzf ~/backups/2025-04-06_19-29-44/pg_wal.tar.gz -C ~/u08_new/
djs10/pg_wal
mkdir -p ~/cje38_new
tar -xzf ~/backups/2025-04-06_19-29-44/16387.tar.gz -C ~/cje38_new/
mkdir -p ~/qdx64_new
tar -xzf ~/backups/2025-04-06_19-29-44/16388.tar.gz -C ~/qdx64_new/
# Создадим в директории кластера файл, сигнализирующий о
восстановлении:
touch ~/u08_new/djs10/recovery.signal
```

Укажем в postgresql.conf команду для загрузки wal файлов:

```
echo "restore_command = 'cp /home/postgres0/u08_new/djs10/pg_wal/%f
%p'" > postgresql.conf
```

Запустим восстановленный кластер: pg_ctl -D /u08_new/djs10 start

Подключимся к бд и посмотрим таблицы:

```
psql -h pg109 -p 9455 -U new_user -d leftbrownmom
\d
```

Этап 4. Логическое повреждение данных

Настраиваем архивирование и добавляем данные

Создадим таблицу с внешними ключами

```
CREATE TABLE IF NOT EXISTS students (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS lessons (
  id SERIAL PRIMARY KEY,
  student_id INTEGER NOT NULL,
  title TEXT NOT NULL,
```

```
  FOREIGN KEY (student_id) REFERENCES students(id)
);
```

Добавим тестовые записи

```
INSERT INTO students (name) VALUES
  ('Азат'),
  ('Ульяна');
INSERT INTO lessons (student_id, title) VALUES
  ((SELECT id FROM students WHERE name='Азат'), 'РСХД азат'),
  ((SELECT id FROM students WHERE name='Ульяна'), 'РСХД ульяна');
```

Посмотрим что получилось:

```
SELECT * from lessons join students on (student_id = students.id);
```

```
leftbrownmom=> SELECT * from lessons join students on (student_id = students.id);
 id | student_id |    title    | id |  name
----+------------+-------------+----+--------
  1 |          1 | РСХД азат   |  1 | Азат
  2 |          2 | РСХД ульяна |  2 | Ульяна
(2 строки)
```

Зафиксируем время изменений

```
leftbrownmom=> select now();
              now
-------------------------------
 2025-04-21 02:04:42.170178+03
(1 строка)
```

Создание дампа

Сделаем скрипт create_dump.sh

```
#!/bin/bash

dumps_dir="${HOME}/dumps/"
dump_name="db-$(date +"%m-%d-%Y-%H-%M-%S").dump"

pg_dump -h pg109 -p 9455 -d leftbrownmom -U new_user -Fc >
$dumps_dir$dump_name
```

Добавим права на исполнение и исполним

```
chmod +x ${HOME}/create_dump.sh
mkdir dumps
bash ./create_dump.sh
```

Иммитация порчи данных

```
ALTER TABLE lessons DROP CONSTRAINT lessons_student_id_fkey;

UPDATE lessons
  SET student_id = 10000
WHERE id IN (
  SELECT id FROM lessons LIMIT 1
);

DELETE FROM students where name = 'Ульяна';
```

Посмотрим что получилось:

```
SELECT * from lessons join students on (student_id = students.id);
```

```
leftbrownmom=> SELECT * from lessons join students on (student_id = students.id);
 id | student_id | title | id | name
----+------------+-------+----+------
(0 строк)
```

Восстановление данных

Создадим скрипт

```
#!/bin/bash

dumps_dir="dumps/"

dump_name=$1

rsync --rsync-path=/usr/local/bin/rsync --archive
postgres0@pg109:~/$dumps_dir$dump_name ~/
pg_restore -h pg114 -p 9455 -d leftbrownmom -U new_user -c
$dump_name -v
rm -rf $dump_name
```

Выдадим разрешения и исполним

```
chmod +x ./restore_dump.sh

bash ./restore_dump.sh db-04-21-2025-02-03-47.dump
```

```
pg_restore: обрабатываются данные таблицы "public.lessons"
pg_restore: обрабатываются данные таблицы "public.students"
pg_restore: обрабатываются данные таблицы "public.test_table1"
pg_restore: из записи оглавления 3782; 0 16394 TABLE DATA test_table1 new_user
pg_restore: ошибка: could not execute query: ������:  ��������� "public.test_table1" ?? ???????????
Выполнялась команда: COPY public.test_table1 (id, data) FROM stdin;
pg_restore: обрабатываются данные таблицы "public.test_table2"
pg_restore: из записи оглавления 3784; 0 16403 TABLE DATA test_table2 new_user
pg_restore: ошибка: could not execute query: ������:  ��������� "public.test_table2" ?? ???????????
Выполнялась команда: COPY public.test_table2 (id, data) FROM stdin;
pg_restore: выполняется SEQUENCE SET lessons_id_seq
pg_restore: выполняется SEQUENCE SET students_id_seq
pg_restore: выполняется SEQUENCE SET test_table1_id_seq
pg_restore: выполняется SEQUENCE SET test_table2_id_seq
pg_restore: создаётся CONSTRAINT "public.lessons lessons_pkey"
pg_restore: создаётся CONSTRAINT "public.students students_pkey"
pg_restore: создаётся CONSTRAINT "public.test_table1 test_table1_pkey"
pg_restore: из записи оглавления 3630; 2606 16401 CONSTRAINT test_table1 test_table1_pkey new_user
pg_restore: ошибка: could not execute query: ������:  ��������� "public.test_table1" ?? ???????????
Выполнялась команда: ALTER TABLE ONLY public.test_table1
    ADD CONSTRAINT test_table1_pkey PRIMARY KEY (id);
```

Посмотрим что мы увидим в таблице

```
SELECT * from lessons join students on (student_id = students.id);
```

```
leftbrownmom=> SELECT * from lessons join students on (student_id = students.id);
 id | student_id |    title    | id |  name
----+------------+-------------+----+--------
  1 |          1 | РСХД азат   |  1 | Азат
  2 |          2 | РСХД ульяна |  2 | Ульяна
(2 строки)
```

Итого: данные удалось восстановить

Вывод

В ходе выполнения данной лабораторной работы мы настроили создание бекапов postgresql, и испытали при различных условиях: полной потери основного узла, повреждении файлов БД или логического повреждения данных.