федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №3

по дисциплине «**Распределенные системы хранения данных**»

Вариант 1

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: P33312

Преподаватель: Шешуков Д. М.

Санкт-Петербург 2023

Постановка задачи и исходные данные

Лабораторная работа включает настройку резервного копирования данных с основного узла на резервный, а также несколько сценариев восстановления. Узел из предыдущей лабораторной работы используется в качестве основного; новый узел используется в качестве резервного. В сценариях восстановления необходимо использовать копию данных, полученную на первом этапе данной лабораторной работы.

Этапы выполнения работы:

Резервное копирование

1. Настроить резервное копирование с основного узла на резервный следующим образом:

Периодические полные копии с помощью SQL Dump.

По расписанию (cron) раз в сутки, методом SQL Dump с сжатием. Созданные архивы должны сразу перемещаться на резервных хост, они не должны храниться на основной системе. Срок хранения архивов на резервной системе - 4 недели. По истечении срока хранения, старые архивы должны автоматически уничтожаться.

2. Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:

Средний объем новых данных в БД за сутки: ~100 МБ.

3. Проанализировать результаты.

Потеря основного узла

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

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:

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

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:

1. В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.

2. Зафиксировать время и симулировать ошибку:

Удалить любые две таблицы (DROP TABLE)

3. Продемонстрировать результат.

4. Выполнить восстановление данных указанным способом.

5. Продемонстрировать и проанализировать результат.

Выполнение

Резервное копирование

Первичная вставка данных в обычное табличное пространство

```
create table t1 as
    (select
        i as first,
        i::text as second,
        i::text as third
    from generate_series(1,1000000) i);

create table t2 as
    (select
        i as first,
        i::text as second,
        now() as third
    from generate_series(1,10000) i);
insert into t2 (first,second,third) select i,i::text,now() from
generate_series(1,1000000) i;
```

```
crazyprog5-# \dt+
                                  Список отношений
 Схема  | Имя |   Тип   | Владелец |  Хранение  | Метод доступа | Размер | Описание
--------+-----+---------+----------+------------+---------------+--------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 49 MB  |
 public | t2  | таблица | postgres2 | постоянное | heap          | 50 MB  |
(2 строки)
```

```
                                Список табличных пространств
    Имя     | Владелец |        Расположение         |     Права доступа     | Параметры | Размер  | Описание
------------+----------+-----------------------------+-----------------------+-----------+---------+----------
 pg_default | postgres2 |                             |                       |           | 133 MB  |
 pg_global  | postgres2 |                             |                       |           | 560 kB  |
 tbstmp1    | postgres2 | /var/db/postgres2/u03/dir7  | postgres2=C/postgres2+|           | 6 bytes |
            |           |                             | s312563=C/postgres2   |           |         |
 tbstmp2    | postgres2 | /var/db/postgres2/u04/dir8  | postgres2=C/postgres2+|           | 6 bytes |
            |           |                             | s312563=C/postgres2   |           |         |
(4 строки)
```

SQL_DUMP

Предварительно добавим резервный узел с основного в число знакомых узлов ssh и сделаем ssh-copy-id, чтобы не вводить пароль при передаче файлов.

```
ssh-keygen
…
ssh-copy-id postgres0@pg120 -i ~/.ssh/id_rsa.pub
```

Имя файла для бекапа генерируется по формату:

```
"backup_`date +"%Y_%m_%d_%H:%M:%S"`"
backup_2023_04_24_10:00:58
```

Для передачи данных на резервный узел бы использовался rsync, так как он значительно быстрее по сравнению с scp, однако узел не предоставляет данный инструмент.

Для того, чтобы не вводить пароль, была добавлена соответствующая запись в .pgpass.

Создание логического бекапа и отправка его на резервный узел:

```
#!/usr/local/bin/bash

BACKUP_FILE=/tmp/pg_crazyprog5_`date +"%Y_%m_%d_%H:%M:%S"`_bak
TARGET_HOST=pg120
TARGET_USER=postgres0
TARGET_PATH=bak

mkdir -p `dirname $BACKUP_FILE`

pg_dump --create \
-d crazyprog5 -h pg107 -p 9035 -U postgres2 -Fc -Z4 -f $BACKUP_FILE

scp $BACKUP_FILE ${TARGET_USER}@${TARGET_HOST}:$TARGET_PATH

rm $BACKUP_FILE
```

Создание задачи cron (запуск каждый день в 2 часа ночи):

```
0 2 * * * /var/db/postgres2/cron_pg_dump.sh
```

Cron устанавливает директорию как обычный пользователь (crontab -e), поэтому он должен прочитать .pgpass.

Удаление на резервном узле каждые 28 дней (также задача cron):

```
#/usr/local/bin/bash

BAK_DIR=/var/db/postgres0/bak

/usr/bin/find $BAK_DIR -name "*" -type f -mtime +28 -exec rm -f {} \;
```

Подсчет размера бекапов

Коэффициент сжатия: 100 / 6.6 = 15

Тогда каждый день размер файла увеличиваться на 100 / 15 = 6.6 Мб

Тогда к концу месяца (30 дней) будет по формуле арифм суммы: 2.5 Гб

Потеря основного узла

Восстановление будет происходить на основании логического бекапа. БД на резервном узле не будет иметь wal файлов изменений/добавлений, а только будет иметь идентичные данные, как и на основном узле.

Устанавливаем PGDATA и инициализируем базу данных

```
export PGDATA="$HOME/u05/dir5"
initdb
```

Создаем директории для табличных пространств:

```
mkdir -p u04/dir8
mkdir -p u03/dir7
```

Создадим БД crazyprog5 и пользователя postgres2:

```
create role postgres2 login password 'postgres' superuser;
create database crazyprog5 with template = template1;
```

Для восстановления требуется воспользоваться командой pg_restore:

```
pg_restore -d crazyprog5 bak/pg_crazyprog5_2023_04_24_11\:35\:45_bak
```

В консоли не высветилось уведомлений, значит все прошло успешно:

```
crazyprog5=# \dt+
                                 Список отношений
 Схема  | Имя |   Тип   | Владелец  |  Хранение  | Метод доступа | Размер | Описание
--------+-----+---------+-----------+------------+---------------+--------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 49 MB  |
 public | t2  | таблица | postgres2 | постоянное | heap          | 50 MB  |
(2 строки)
```

Повреждение файлов БД

По oid находим местоположение таблицы:

```
[postgres2@pg107 ~/u05/dir5]$ find . -name '*16747*'
./base/16397/16747
./base/16397/16747_vm
./base/16397/16747_fsm
[postgres2@pg107 ~/u05/dir5]$
```

Удаляем файл с соответствующим именем.

Данные из таблицы были удалены:

```
crazyprog5=# \dt+
                                 Список отношений
 Схема  | Имя |   Тип   | Владелец  |  Хранение  | Метод доступа | Размер | Описание
--------+-----+---------+-----------+------------+---------------+--------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 48 kB  |
 public | t2  | таблица | postgres2 | постоянное | heap          | 50 MB  |
(2 строки)
```

```
crazyprog5=# select * from t1;
ОШИБКА:  не удалось открыть файл "base/16397/16747": No such file or directory
```

Изменяем исходное расположение PGDATA и пересоздаем там базу данных.

Сначала передаем бекап на основную машину:

```
scp bak/pg_crazyprog5_2023_04_24_11\:35\:45_bak postgres2@pg107
```

Изменяем директорию PGDATA:

```
export PGDATA=/var/db/postgres2/u06/dir5
```

Запускаем initdb.

Запускаем новый инстанс, создаем бд, делаем pg_restore:

```
pg_ctl start
create database crazyprog5 with template = template1;
pg_restore pg_crazyprog5_2023_04_24_11\:35\:45_bak -d crazyprog5 -p 9035
```

Меняем конфигурационные файлы:

```
cp u05/dir5/pg_hba.conf u05/dir5/postgresql.conf u06/dir5/
```

Останавливаем старый инстанс, перезапускаем новый:

```
PGDATA=/var/db/postgres2/u05/dir5 pg_ctl stop -mf
pg_ctl stop -mf
pg_ctl start
```

```
crazyprog5=# \dt+
                                  Список отношений
 Схема  | Имя |   Тип   | Владелец  |  Хранение  | Метод доступа | Размер | Описание
--------+-----+---------+-----------+------------+---------------+--------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 49 MB  |
 public | t2  | таблица | postgres2 | постоянное | heap          | 50 MB  |
(2 строки)

crazyprog5=# select * from t1;
  first  | second  |  third
---------+---------+---------
      1 | 1       | 1
      2 | 2       | 2
```

Теперь снова можем получить доступ к данным в таблице t1.

Логическое повреждение данных

Для демонстрации создадим 3 новых таблицы, добавим в них несколько строчек данных.

```
create table t3 (id serial primary key, name varchar(255) not null unique);
create table t4 (id serial primary key, intval integer not null);
create table t5 (id serial primary key, boolval boolean not null);
```

```
crazyprog5=# select * from t3;
 id | name
----+------
  1 | a
  2 | b
(2 строки)

crazyprog5=# select * from t4;
 id | intval
----+--------
  3 |     10
  4 |    100
(2 строки)

crazyprog5=# select * from t5;
 id | boolval
----+---------
  5 | t
  6 | f
(2 строки)
```

Содержимое созданных таблиц.

Теперь создадим backup, добавим его на резервный узел и произведем restore с помощью backup файла bak/pg_crazyprog5_2023_04_24_12\:52\:22_bak.

```
                          Список отношений
 Схема  | Имя |   Тип   | Владелец |  Хранение  | Метод доступа |   Размер   | Описание
--------+-----+---------+----------+------------+---------------+------------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 98 MB      |
 public | t2  | таблица | postgres2 | постоянное | heap          | 101 MB     |
 public | t3  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
 public | t4  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
 public | t5  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
(5 строк)
```

Вернемся к основному узлу, добавим несколько записей в таблицы и удалим «по ошибке» таблицы t3, t4.

```
crazyprog5=# insert into t3 values (3, 'c'), (4, 'd'), (5, 'e');
INSERT 0 3
crazyprog5=# insert into t4 values (5, 1000), (6, 1001);
INSERT 0 2
crazyprog5=# insert into t5 values (7, true), (8, true);
INSERT 0 2
crazyprog5=# drop table t3, t4;
DROP TABLE
```

```
crazyprog5=# select * from t5;
 id | boolval
----+---------
  5 | t
  6 | f
  7 | t
  8 | t
(4 строки)
```

Выполним восстановление данных (восстановим таблицы t3, t4):

```
scp bak/pg_crazyprog5_2023_04_24_12\:52\:22_bak postgres2@pg107
```

Дропнем таблицы t3, t4, t5 и восстановим их из бекапа:

```
pg_restore --clean -p 9035 -d crazyprog5 pg_crazyprog5_2023_04_24_12\:52\:22_bak
```

Посмотрим какие данные теперь хранятся в бд:

```
crazyprog5=# \dt+
                                  Список отношений
 Схема  | Имя |   Тип   | Владелец |  Хранение  | Метод доступа |   Размер   | Описание
--------+-----+---------+----------+------------+---------------+------------+----------
 public | t1  | таблица | postgres2 | постоянное | heap          | 49 MB      |
 public | t2  | таблица | postgres2 | постоянное | heap          | 50 MB      |
 public | t3  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
 public | t4  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
 public | t5  | таблица | postgres2 | постоянное | heap          | 8192 bytes |
(5 строк)
```

```
crazyprog5=# select * from t3;
 id | name
----+------
  1 | a
  2 | b
(2 строки)

crazyprog5=# select * from t4;
 id | intval
----+--------
  3 |     10
  4 |    100
(2 строки)

crazyprog5=# select * from t5;
 id | boolval
----+---------
  5 | t
  6 | f
(2 строки)
```

Как видим, последние изменения, которые не были забекаплены, не были восстановлены, то есть данные частично потеряны.

Вывод

В ходе выполнения данной лабораторной работы были разобраны способы создания бекапов, отличие логических и физический бекапов, предназначение wal файлов, а также возможности востановления рабостоспособности базы данных при помощи логического бекапа в случае различных сбоев.

Кроме того, изучено как пользоваться утилитой cron.