Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине

‘Распределённые системы хранения данных’

Вариант №282512

*Выполнил:*

Студент группы P33302

Тюрин Святослав Вячеславович

*Преподаватель:*

Шешуков Дмитрий Михайлович



Санкт-Петербург, 2024

**Задание:**

### **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: 650МБ.
  + Средний объем измененных данных за сутки: 150МБ.
* 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 любого табличного пространства 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:
  + перезаписать строки любой таблицы “мусором” (INSERT, UPDATE)
* Продемонстрировать результат.
* Выполнить восстановление данных указанным способом.
* Продемонстрировать и проанализировать результат.

**Выполнение:**

### **1. Резервное копирование**

ssh -J s282509@helios.cs.ifmo.ru:2222 postgres0@pg134

TLW6n4pR

ssh -J s282509@helios.cs.ifmo.ru:2222 postgres1@pg136

o2ASgdRS

Создаем пользователя для будущих реплик:

create role replica replication login password '1234';



postgesql.conf



Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Основной узел:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Резервный узел:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

На основном узле создадим cron-файл через команду (crontab -e), в котором опишем правило для запуска нашего скрипта раз в неделю по воскресеньям в 00:00:00. Проверим список запланированных задач:



Проверим работу скрипта:

Изображение выглядит как текст, снимок экрана, черный, Шрифт

Автоматически созданное описание

Резервная копия создалась:



**Расчет объема:**

Размер одного бэкапа (изначально):



Количество данных для перезаписывания: 650+150 Мб в день. Копии создаются 1 раз в неделю, следовательно за месяц будет создано 4 копии. Так как на основном узле копии хранятся 7 дней, а на резервном 28 дней, то на момент прохождения месяца на основном будет 1 копия, а на резервном – 4.

Посчитаем с помощью арифметической прогрессии размер копий за 28 дней:

S1 = (9.9\*2+650(7-1))/2\*7 = 13719 Мб

S2 =9.9\*2+650(14-1))/2\*14 = 59289 Мб

S3 =9.9\*2+650(21-1))/2\*21 = 136708 Мб

S3 =9.9\*2+650(28-1))/2\*28 = 245977 Мб

Сумма = 455693 Мб = 445 Гб

Так как wal файлы у нас архивируются, то их объем не будет превышать 1Гб (max\_wal\_size)

Общий размер = 445 +4\*1 = 449 Гб

**Потеря основного узла**

Воссоздадим файловую структуру кластера для восстановления:

mkdir -p $HOME/ivo23

chmod 700 ivo23/

cp -r ~/backups/backup\_20240508233840/\* ~/ivo23/

Файлы табличного пространства

mkdir -p $HOME/kln97

chmod 700 kln97/

cp -r ~/backups/backup\_20240508233840/pg\_tblspc/16447/ ~/kln97/

Меняем символические ссылки:

ln -s /var/db/postgres1/kln97/ ~/ivo23/pg\_tblspc/16447

Укажем в postgresql.conf команду для загрузки wal файлов:

restore\_command = 'cp /var/db/postgres1/wal\_archive/%f %p'

Создадим в директории кластера файл, сигнализирующий о восстановлении:

touch ~/ivo23/recovery.signal

Запускаем резервный кластер: postgres -D $HOME/ivo23 >~/logfile 2>&1 &

Проверяем работоспособность: psql -h localhost -p 9500 -U postgres0 sickredmath

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Анализ выполнения: Восстановление завершилось успешно.

**Повреждение файлов БД**

Я запустил кластер с одного из бэкапов и удалил табличное пространство.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

При попытке работать с табличным пространством – возникают ошибки



Перенесём копию на основной узел:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Запускаем поврежденный кластер в режиме восстановления wal-файлов и указываем команду для восстановления

Меняем команду восстановления в файле postgresql.conf для текущего бэкапа:



Меняем символическую ссылку:

ln -s /var/db/postgres0/ivo23\_new/ /var/db/postgres0/backups/BACKUP\_2024-05-08-23\:38\:40/pg\_tblspc/16447

Запускаем в режиме восстановления

touch ~/backups/BACKUP\_2024-05-08-23\:38\:40/recovery.signal

postgres -D \$HOME/backups/BACKUP\_2024-05-08-23\:38\:40/ >~/logfile 2>&1 &

Результат:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Анализ

Восстановление завершилось успешно и работа с табличным пространством корректно.

**Логическое повреждение данных**

Сначала БД будет наполнена новыми данными, с которыми потом будут производиться неаккуратные действия.

Изображение выглядит как текст, снимок экрана, Шрифт

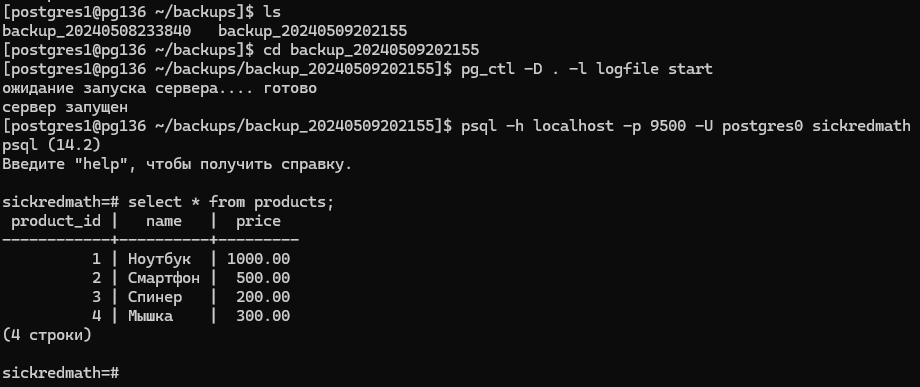
Автоматически созданное описание

После этого сделаем РК и отправим её на резервный узел:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

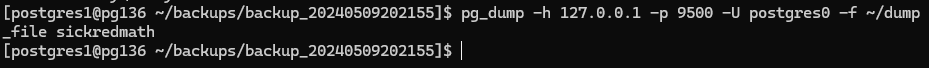
Автоматически созданное описание

Далее после запуска на резервном узле появятся следующие данные и та самая пока не «испорченная» БД (данные) на которой будет сделан дамп:



Далее делается сам дамп БД sickredmath, которая будет испорчена на основном узле:

pg\_dump -h 127.0.0.1 -p 9500 -U postgres0 -f ~/dump\_file sickredmath



Удалим таблицу с продуктами на основном узле:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Теперь нужно скопировать дамп с резервного узла и применить его на основном:

scp dump\_file postgres0@pg134:~/

Далее восстанавливаем данные из файла в текстовом формате:  
psql -h 127.0.0.1 -p 9500 -d sickredmath < ~/dump\_file

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Видно, что таблица опять появилась с теми же данными.

**Выводы**

Во время выполнения лабораторной работы я изучил способы непрерывного бекапа кластера PostgreSQL, на практике настроил и применил его при различных сбоях.