Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

Администрирование СУБД

Лабораторная работа №3

Вариант №347385

Группа: P3324

Выполнил: Маликов Глеб Игоревич

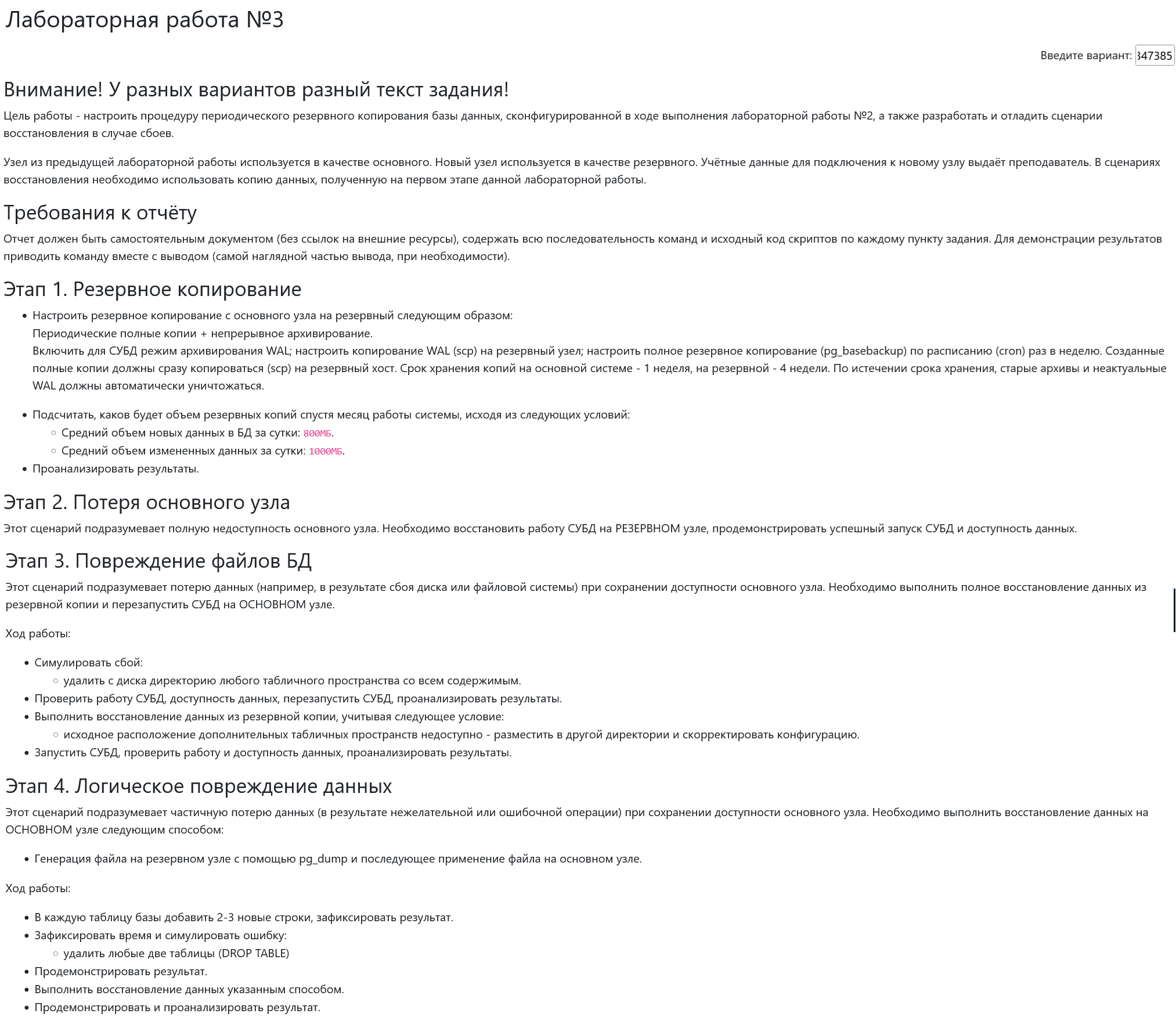
Преподаватель:

Николаев Владимир Вячеславович

Санкт-Петербург

2024г.

# Задание



# Реализация

## Этап 1. Резервное копирование

### Включение режима архивирования WAL на основном узле

Изменяем конфигурационный файл postgresql.conf на основном узле, отправляя WAL-логи на резервный узел:

Создадим директорию для WAL файлов:

wal\_level = replica

archive\_mode = on

archive\_command = 'scp %p postgres0@pg175:~/wal\_archive/%f'

В резервном узле создаем директорию для хранения WAL:

mkdir -p wal\_archive # учитываем, что директория будет создана в домашнем каталоге пользователя

На основном узле сгенерируем SSH-ключ для автоматической авторизации scp:

ssh-keygen -t rsa -b 4096 -C "postgres1@pg167" # на все вопросы отвечаем Enter

ssh-copy-id -i ~/.ssh/id\_rsa.pub postgres0@pg175

Проверка: (доступ к резервному узлу без пароля)

[postgres1@pg167 ~]$ ssh postgres0@pg175

Last login: Sat Nov  2 12:50:24 2024 from 192.168.11.167

[postgres0@pg175 ~]$

Загружаем изменения в конфигурацию:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" main\_pg167/postgresql.conf postgres1@pg167:~/khk43

### Настройка полного резервного копирования (pg\_basebackup) по расписанию

Создаем директорию для хранения резервных копий на основном узле, также на резервном узле:

mkdir -p ~/backups

Для разрешения подключения для репликации на основном узле добавим строку в pg\_hba.conf

local   replication     all                                     peer map=my\_map

Загрузим изменения:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" main\_pg167/pg\_hba.conf postgres1@pg167:~/khk43

Перезапустим сервер:

pg\_ctl -D ~/khk43 restart

Создаем скрипт(./main\_pg167/backup.sh) для резервного копирования `backup.sh` на основном узле:

#!/bin/sh

CURRENT\_DATE=$(date "+%Y-%m-%d\_%H:%M:%S")

BACKUP\_DIR="~/backups/$CURRENT\_DATE"

mkdir -p "$BACKUP\_DIR"

# Создаем полную резервную копию

pg\_basebackup -D "$BACKUP\_DIR" -F tar -z -P -p 9555 # 9555 - порт указанный в основном узле postgresql.conf

# Копируем резервную копию на резервный узел

scp "$BACKUP\_DIR"/\*.tar.gz postgres0@pg175:~/backups/

# Удаляем резервные копии старше 7 дней на основном узле

find ~/backups/ -type d -mtime +7 -exec rm -rf {} \;

# Удаляем WAL-файлы старше 7 дней на основном узле

find ~/oka84/ -type f -mtime +7 -exec rm -f {} \; # по предыдущему заданию WAL-файлы хранятся в '~/oka84/'

# Удаляем резервные копии старше 28 дней на резервном узле

ssh postgres0@pg175 'find ~/backups/ -type d -mtime +28 -exec rm -rf {} \;'

# Удаляем WAL-файлы старше 28 дней на резервном узле

ssh postgres0@pg175 'find ~/wal\_archive/ -type f -mtime +28 -exec rm -f {} \;'

Загружаем скрипт на основной узел:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" main\_pg167/backup.sh postgres1@pg167:~

Сделаем скрипт исполняемым:

chmod +x backup.sh

Добавляем задачу в cron на основном узле:

crontab -e

Добавляем строку (каждое воскресенье в 2 часа ночи):

0 2 \* \* 0 ~/backup.sh >> ~/backup.log 2>&1

Проверим работоспособность скрипта:

bash ~/backup.sh >> ~/backup.log 2>&1

Результаты выполнения скрипта:

[postgres1@pg167 ~]$ cat backup.log

ожидание контрольной точки

    2/31300 КБ (0%), табличное пространство 0/3

    2/31300 КБ (0%), табличное пространство 1/3

    4/31300 КБ (0%), табличное пространство 1/3

    4/31300 КБ (0%), табличное пространство 2/3

31314/31314 КБ (100%), табличное пространство 2/3

31314/31314 КБ (100%), табличное пространство 3/3

### Подсчет объема резервных копий

**Исходные данные**

* Средний объем новых данных в БД за сутки: 800 МБ
* Средний объем измененных данных за сутки: 1000 МБ
* Частота полного резервного копирования: раз в неделю (посредством pg\_basebackup).
* Архивирование WAL: непрерывное, ежедневно.
* Срок хранения копий:
  + На основном узле: 1 неделя.
  + На резервном узле: 4 недели.

**Подсчет**

Объем данных за неделю:

* Новые данные за неделю: 800 МБ/сутки \* 7 дней = 5.6 ГБ.
* Измененные данные за неделю (WAL): 1000 МБ/сутки \* 7 дней = 7 ГБ.

Объем резервных копий на основном узле:

* Полные резервные копии:
  + Количество хранимых копий: 1 (поскольку срок хранения — 1 неделя).
  + Объем последней полной копии: 5.6 ГБ \* 4 недели = 22.4 ГБ. (Предполагается, что каждая последующая полная копия включает накопленные данные за предыдущие недели).
* - Архивы WAL:
  + Количество хранимых архивов: 1 неделя.
  + Общий объем WAL за неделю: 7 ГБ.
* Итого объем на основном узле = 22.4 ГБ (полные копии) + 7 ГБ (WAL) = 29.4 ГБ.

Объем резервных копий на резервном узле:

* Полные резервные копии:
  + Количество хранимых копий: 4 (поскольку срок хранения — 4 недели).
  + Общий объем полных копий за месяц: ГБ.
* Архивы WAL:
  + Количество хранимых архивов: 4 недели.
  + Общий объем WAL за месяц: 4 \* 7 = 28 ГБ.
* Итого объем на резервном узле = 56 ГБ (полные копии) + 28 ГБ (WAL) = 84 ГБ.

Тем не менее из-за сжатия данных, объем резервных копий будет меньше.

## Этап 2. Потеря основного узла

Для будущей проверки добавим таблицу в базу данных на основном узле:

CREATE TABLE test\_table (id SERIAL PRIMARY KEY, data TEXT);

INSERT INTO test\_table (data) VALUES ('Hello, World!');

postgres=# CREATE TABLE test\_table (id SERIAL PRIMARY KEY, data TEXT);

INSERT INTO test\_table (data) VALUES ('Hello, World!');

CREATE TABLE

INSERT 0 1

postgres=# SELECT \* FROM test\_table;

 id |     data

----+---------------

  1 | Hello, World!

(1 строка)

Выполним резервное копирование на основном узле:

bash ~/backup.sh >> ~/backup.log 2>&1

Установим restore\_command в postgresql.conf (./reserve\_pg175/postgresql.conf) на резервном узле:

restore\_command = 'cp /var/db/postgres0/wal\_archive/%f "%p"'

В резервном узле создадим скрипт для восстановления базы данных restore.sh(./reserve\_pg175/restore.sh):

#!/bin/sh

mkdir -p ~/khk43 #  новый каталог для PostgreSQL, как в предыдущей лабораторной работе

echo "Создан каталог ~/khk43"

cd ~/backups

BACKUP\_DIR=$(ls -td \*/ | head -n 1) # выбираем последнюю резервную копию

cd $BACKUP\_DIR

echo "Выбрана резервная копия $BACKUP\_DIR"

tar -xzf base.tar.gz -C ~/khk43

tar -xzf pg\_wal.tar.gz -C ~/khk43/pg\_wal

echo "Распакована резервная копия"

# создание директорий для tablespaces по аналогии с предыдущей лабораторной работой

mkdir -p ~/mqb89

mkdir -p ~/utr38

echo "Созданы каталоги для tablespaces"

tar -xzf 16384.tar.gz -C ~/mqb89 # OID табличного пространства mqb89

tar -xzf 16385.tar.gz -C ~/utr38 # OID табличного пространства utr38

echo "Распакованы табличные пространства"

chown -R postgres0 ~/khk43

chmod -R 750 ~/khk43 ~/mqb89 ~/utr38 # Маска прав должна быть u=rwx (0700) или u=rwx,g=rx (0750).

chown -R postgres0 ~/mqb89

chown -R postgres0 ~/utr38

echo "Установлены права доступа"

touch ~/khk43/recovery.signal

chown postgres0 ~/khk43/recovery.signal

chmod -R 700 ~/khk43/recovery.signal

echo "Создан файл recovery.signal"

cd ~

cp ~/pg\_hba.conf ~/khk43

cp ~/postgresql.conf ~/khk43

cp ~/pg\_ident.conf ~/khk43

echo "Скопированы .conf файлы"

# запуск PostgreSQL

pg\_ctl -D ~/khk43 -l файл\_журнала start

# Остановка PostgreSQL

pg\_ctl -D ~/khk43 stop

# Изменяем символические ссылки на табличные пространства

cd ~/khk43/pg\_tblspc

rm 16384

rm 16385

ln -s ~/mqb89 16384

ln -s ~/utr38 16385

cd ~

# Запуск PostgreSQL

pg\_ctl -D ~/khk43 -l файл\_журнала start

Загружаем скрипт и postgresql.conf на резервный узел:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" reserve\_pg175/restore.sh postgres0@pg175:~

scp -o "ProxyJump s372819@se.ifmo.ru:2222" reserve\_pg175/postgresql.conf postgres0@pg175:~

Для возможности повторения сценария cleanup.sh(./reserve\_pg175/cleanup.sh) на резервном узле:

#!/bin/sh

pg\_ctl -D ~/khk43 stop

# Копирование .conf файлов

cp ~/khk43/postgresql.conf ~/khk43/pg\_hba.conf ~/khk43/pg\_ident.conf ~

rm -rf ~/khk43

rm -rf ~/mqb89

rm -rf ~/utr38

Загружаем скрипт на резервный узел:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" reserve\_pg175/cleanup.sh postgres0@pg175:~

chmod +x cleanup.sh

Добавим в pg\_ident.conf(./reserve\_pg175/pg\_ident.conf) на резервном узле:

my\_map          postgres0               postgres1

Применим изменения:

pg\_ctl -D ~/khk43 restart

Для проверки целостности данных на резервном узле выполним:

psql -p 9555 -d postgres -U postgres1

SELECT \* FROM test\_table;

Результат:

postgres=# SELECT \* FROM test\_table;

 id |     data

----+---------------

  1 | Hello, World!

(1 строка)

## Этап 3. Повреждение файлов БД

### Симулирование сбоя

Для будущей проверки добавим таблицу в табличное пространство на основном узле:

CREATE TABLE test\_table\_mqb89 (id SERIAL PRIMARY KEY, data TEXT) TABLESPACE mqb89;

INSERT INTO test\_table\_mqb89 (data) VALUES ('Data in tablespace mqb89');

SELECT \* FROM test\_table\_mqb89;

postgres=# SELECT \* FROM test\_table\_mqb89;

 id |           data

----+--------------------------

  1 | Data in tablespace mqb89

(1 строка)

Пусть у нас выполнилось сохранение данных на основном узле:

bash ~/backup.sh >> ~/backup.log 2>&1

Симулируем сбой на основном узле, удалив директорию с табличным пространством:

rm -rf ~/mqb89

### Проверка работоспособности

Проверяем работу СУБД, пытаясь получить данные из таблицы ts\_table:

psql -p 9555 -d postgres -U postgres1

SELECT \* FROM test\_table\_mqb89;

Результат:

postgres=# SELECT \* FROM test\_table\_mqb89;

ОШИБКА:  не удалось открыть файл "pg\_tblspc/16384/PG\_16\_202307071/5/16430": No such file or directory

postgres=#

Попробуем перезапустить СУБД:

pg\_ctl -D ~/khk43 stop

pg\_ctl -D ~/khk43 -l файл\_журнала start

Результат:

[postgres1@pg167 ~]$ psql -p 9555 -d postgres -U postgres1

psql (16.4)

Введите "help", чтобы получить справку.

postgres=# SELECT \* FROM test\_table\_mqb89;

ОШИБКА:  не удалось открыть файл "pg\_tblspc/16384/PG\_16\_202307071/5/16430": No such file or directory

postgres=#

Как видим, СУБД смогла перезапуститься, так как PostgreSQL не требует наличия всех табличных пространств для запуска. Главное, чтобы не было ошибок в системных таблицах.

Тем не менее не смогла получить данные из таблицы, так как файлы табличного пространства были утеряны.

### Восстановление данных

Учитывая, что исходное расположение табличного пространства недоступно, разместим его в другой директории и скорректируем конфигурацию.

recover\_mqb89.sh(./main\_pg167/recover\_mqb89.sh):

pg\_ctl -D ~/khk43 stop

mkdir -p ~/new\_mqb89

echo "Создана директория ~/new\_mqb89"

cd ~/backups

BACKUP\_DIR=$(ls -td \*/ | head -n 1) # выбираем последнюю резервную копию

cd $BACKUP\_DIR

echo "Выбрана резервная копия $BACKUP\_DIR"

tar -xzf 16384.tar.gz -C ~/new\_mqb89

echo "Распаковано табличное пространство"

chown -R postgres1 ~/new\_mqb89

chmod 750 ~/new\_mqb89

echo "Установлены права доступа"

cd ~/khk43/pg\_tblspc

rm 16384

ln -s ~/new\_mqb89 16384

echo "Изменены символические ссылки"

cd ~

pg\_ctl -D ~/khk43 -l файл\_журнала start

Загружаем скрипт на основной узел:

scp -o "ProxyJump s372819@se.ifmo.ru:2222" main\_pg167/recover\_mqb89.sh postgres1@pg167:~

Сделаем скрипт исполняемым:

chmod +x recover\_mqb89.sh

Выполним скрипт на основном узле:

bash ~/recover\_mqb89.sh

Проверим работоспособность СУБД:

psql -p 9555 -d postgres -U postgres1

SELECT \* FROM test\_table\_mqb89;

Результат:

[postgres1@pg167 ~]$ psql -p 9555 -d postgres -U postgres1

psql (16.4)

Введите "help", чтобы получить справку.

postgres=# SELECT \* FROM test\_table\_mqb89;

 id |           data

----+--------------------------

  1 | Data in tablespace mqb89

(1 строка)

postgres=#

Как видим, данные восстановлены успешно.

## Этап 4. Логическое повреждение данных

### Добавление новых строк в таблицы

В основном узле добавим новые строки в таблицы:

INSERT INTO test\_table (data) VALUES ('New data 1');

INSERT INTO test\_table (data) VALUES ('New data 2');

INSERT INTO test\_table\_mqb89 (data) VALUES ('New data 1');

INSERT INTO test\_table\_mqb89 (data) VALUES ('New data 2');

SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

Результат:

postgres=# SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

 id |     data

----+---------------

  1 | Hello, World!

  2 | New data 1

  3 | New data 2

(3 строки)

 id |           data

----+--------------------------

  1 | Data in tablespace mqb89

  2 | New data 1

  3 | New data 2

(3 строки)

Так как требуется генерация файла на резервном узле с помощью pg\_dump, то там должна быть актуальная копия данных. Поэтому выполним вышеописанный скрипт для резервного копирования на резервном узле:

bash ~/backup.sh >> ~/backup.log 2>&1

### Симулирование ошибки

Зафиксируем время и симулируем ошибку, удалив таблицы в основном узле:

psql -p 9555 -d postgres -U postgres1

SELECT now();

Результат:

postgres=# SELECT now();

              now

-------------------------------

 2024-11-03 16:41:29.585481+03

(1 строка)

DROP TABLE test\_table;

DROP TABLE test\_table\_mqb89;

Демонстрация ошибки:

SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

Результат:

postgres=# SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

ОШИБКА:  отношение "test\_table" не существует

СТРОКА 1: SELECT \* FROM test\_table;

                        ^

ОШИБКА:  отношение "test\_table\_mqb89" не существует

СТРОКА 1: SELECT \* FROM test\_table\_mqb89;

                        ^

### Восстановление данных

В резервном узле будет запущен скрипт restore.sh(./reserve\_pg175/restore.sh) для восстановления данных. (Вместе с cleanup.sh(./reserve\_pg175/cleanup.sh)).

Подразумевается, что ранее резервное копирование было выполнено успешно.

./cleanup.sh

./restore.sh

Таким образом, на резервном узле будут актуальные данные:

psql -p 9555 -d postgres -U postgres1

SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

Результат:

postgres=# SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

 id |     data

----+---------------

  1 | Hello, World!

  2 | New data 1

  3 | New data 2

(3 строки)

 id |           data

----+--------------------------

  1 | Data in tablespace mqb89

  2 | New data 1

  3 | New data 2

(3 строки)

На резервном узле сгенерируем файл с помощью pg\_dump:

pg\_dump -p 9555 -U postgres1 -d postgres > ~/pg\_dump.sql

На резервном узле сгенерируем SSH-ключ для автоматической авторизации scp:

ssh-keygen -t rsa -b 4096 -C "postgres0@pg175" # на все вопросы отвечаем Enter

ssh-copy-id -i ~/.ssh/id\_rsa.pub postgres1@pg167

Загрузим файл дампа на основной узел из резервного:

scp ~/pg\_dump.sql postgres1@pg167:~

Для возможности повторения сценария generate\_and\_send\_dump.sh(./reserve\_pg175/generate\_and\_send\_dump.sh) на резервном узле:

#!/bin/sh

# Ожидаем что существует актуальная резервная копия в '~/backups'

cd ~

./cleanup.sh

./restore.sh

# Генерация дампа

pg\_dump -p 9555 -U postgres1 -d postgres > ~/pg\_dump.sql

echo "Сгенерирован дамп"

# Отправка дампа на основной узел

scp ~/pg\_dump.sql postgres1@pg167:~

echo "Дамп отправлен"

В основном узле восстановим данные из дампа:

psql -p 9555 -d postgres -U postgres1 -f ~/pg\_dump.sql

Проверим, что данные восстановлены:

psql -p 9555 -d postgres -U postgres1

SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

Результат:

[postgres1@pg167 ~]$ psql -p 9555 -d postgres -U postgres1

psql (16.4)

Введите "help", чтобы получить справку.

postgres=# SELECT \* FROM test\_table;

SELECT \* FROM test\_table\_mqb89;

 id |     data

----+---------------

  1 | Hello, World!

  2 | New data 1

  3 | New data 2

(3 строки)

 id |           data

----+--------------------------

  1 | Data in tablespace mqb89

  2 | New data 1

  3 | New data 2

(3 строки)

postgres=#

# Заключение

В ходе выполнения лабораторной работы были рассмотрены и реализованы этапы настройки резервного копирования, восстановления данных при потере основного узла, восстановления данных при повреждении файлов БД, восстановления данных при логическом повреждении.