Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина *«Распределенные системы хранения данных»*

Отчет по лабораторной работе №3

Вариант: 368122

Студент:

Барсуков Максим Андреевич,
группа P3315

Преподаватель:

Харитонова Анастасия Евгеньевна

г. Санкт-Петербург, 2025 г.

Оглавление

Задание

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 выдает 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 с помощью SQL Dump.
  По расписанию (cron) раз в сутки, методом SQL Dump с сжатием. Созданные архивы должны сразу перемещаться на резервный 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.
- 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: 350МБ.
- 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:

- Восстановление с использованием архивных WAL файлов. (СУБД должна работать в режиме архивирования WAL, потребуется задать параметры восстановления).

Ход работы:

- В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
- Зафиксировать время и симулировать ошибку:
    - удалить каждую вторую строку в любой таблице (DELETE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

Выполнение

Узлы:

- Основной: `postgres1@pg186`
- Резервный: `postgres2@pg185`

Создадим суперпользователя:

```
CREATE ROLE admin SUPERUSER CREATEDB CREATEROLE LOGIN PASSWORD 'admin';
```

Этап 1. Резервное копирование

Напишем bash-скрипт, который будет создавать дамп кластера, отправлять его на резервный узел, после удаленно удалять устаревшие копии.

Основной узел:

```
#!/usr/bin/env bash

DATE=$(date +%Y-%m-%d-%H-%M-%S)

# Создание дампа
pg_dumpall -h localhost -p 9114 -U postgres1 | gzip -9 >
~/backups/pgdump-$DATE.sql.gz

# Перемещение на резервный хост
scp ~/backups/pgdump-$DATE.sql.gz postgres2@pg185:~/backups

# Удаление локальной копии
rm ~/backups/pgdump-$DATE.sql.gz

# Очистка старых архивов на резервном хосте
ssh postgres2@pg185 "find ~/backups -name '*.sql.gz' -mtime +28 -delete"
```

На основном узле создадим cron-файл через команду (`crontab -e`), в котором опишем правило для запуска нашего скрипта каждые сутки в 2:00.

```
0 2 * * * ~/backups/backup.sh
```

После чего запустим с помощью команды. Проверим список запланированных задач.

```
[postgres1@pg186 ~/backups]$ crontab -l
0 2 * * * ~/backups/backup.sh
```

Чтобы скрипт работал корректно, добавим ssh ключи для доступа к резервному узлу без пароля:

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub postgres2@pg185
```

Проверим работоспособность:

Основной узел **pg186**:

```
[postgres1@pg186 ~]$ ./backups/backup.sh
pgdump-2025-04-21-01-24-08.sql.gz                                    100% 2981     3.9MB/s   00:00
```

Резервный узел **pg185**:

```
[postgres2@pg185 ~]$ ls backups/
pgdump-2025-04-21-01-24-08.sql
```

Как и ожидалось, сжатый дамп БД был перенесен на резервный узел.

Расчет объема резервных копий

Исходный размер 4,5K, можем принять за 0, так как незначителен при заданном объеме новых данных в сутки.

```
[postgres2@pg185 ~]$ du -h backups/fakebrownroad-2025-04-20-23-24-57.dump
4,5K    backups/fakebrownroad-2025-04-20-23-24-57.dump
```

Условия

- Ежедневные полные SQL-дампы с использованием сжатия.
- Срок хранения: 30 дней.
- Время жизни дампа: 4 недели (28 дней).
- Рост данных:
    - Новые данные: 650 МБ/сутки.
    - Измененные данные: 350 МБ/сутки (не увеличивают общий объем, только модифицируют существующие).
- Начальный размер базы: 0 МБ (для простоты расчёта).
- Коэффициент сжатия: 3:1 (примерно 33% от исходного размера, использован уровень сжатия 9).

Вычисления

В 30-й день будут храниться дампы с 3 по 30-й день.

Размер базы на день t: $S(t) = 650 \cdot t$ МБ.

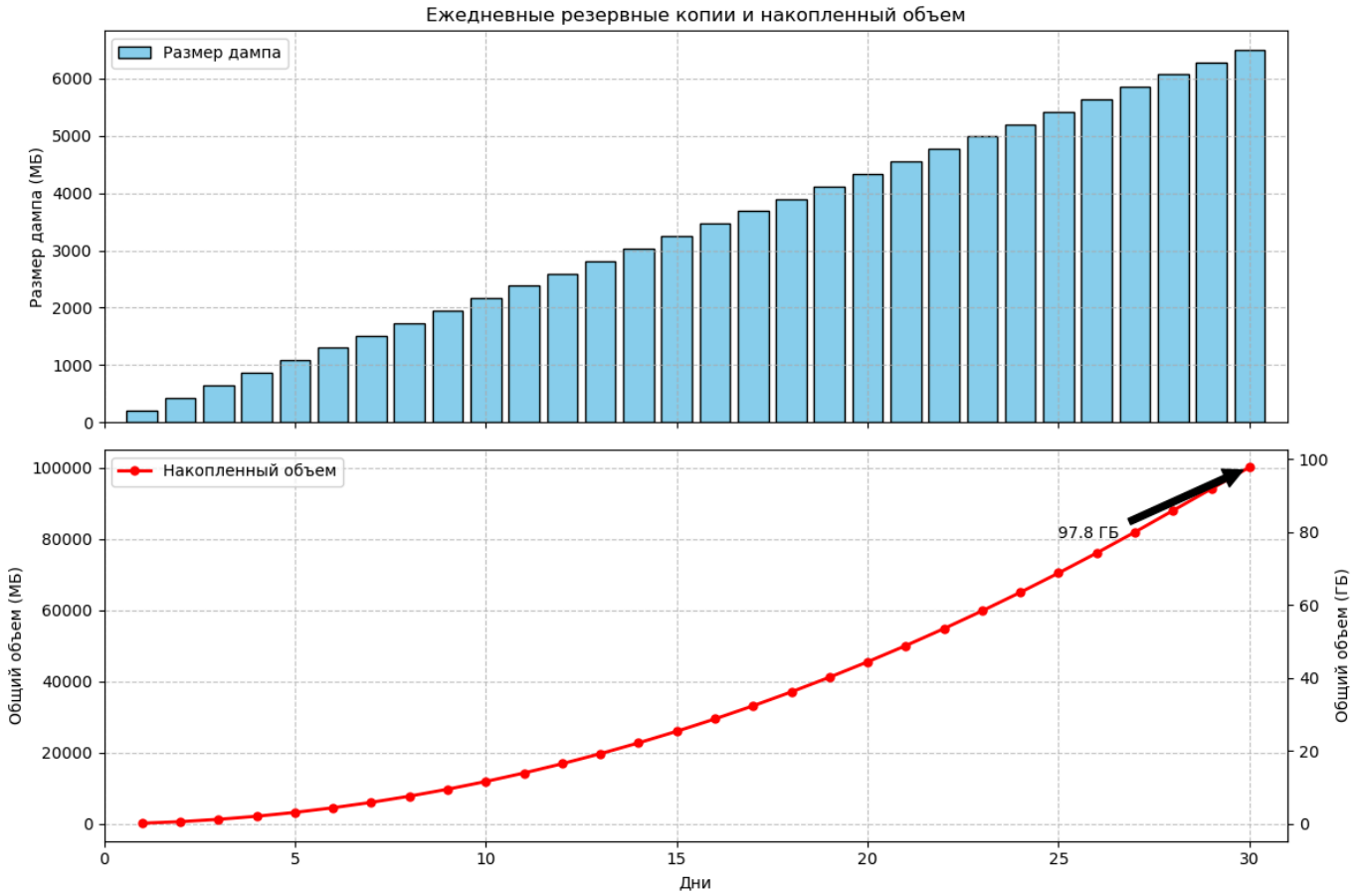
Размер сжатого дампа на день t: $D(t) = \frac{650 \cdot t}{3}$ МБ.

Общий объем на 30-й день:

$$Total = \sum_{t=3}^{30} D(t) = \frac{650}{3} \cdot \sum_{t=3}^{30} t = \frac{650}{3} \cdot \left(\frac{30 \cdot 31}{2} - \frac{2 \cdot 3}{2}\right) = \frac{650}{3} \cdot 462 \approx 100100 \text{ МБ} \approx 98 \text{ ГБ}$$

Анализ

Дамп на 30-й день: 6.5 ГБ.
Средний размер дампа: ≈3.28 ГБ/день.
Пиковый объём: 6.5 ГБ (последний день).



Этап 2. Потеря основного узла

Имитируем полную недоступность основного узла остановкой PostgreSQL:

```
pg_ctl -D $HOME/ubi26 stop
```

```
[postgres1@pg186 ~]$ pg_ctl -D $HOME/ubi26 stop
ожидание завершения работы сервера.... готово
сервер остановлен
[postgres1@pg186 ~]$
```

Восстановление данных

Создадим аналогичную БД на резервном узле:

```
export PGDATA=$HOME/ubi26
export PGWAL=$PGDATA/pg_wal
export PGLOCALE=ru_RU.UTF-8
export PGENCODE=UTF8
export PGUSERNAME=postgres1
export PGHOST=pg185
export LANG=ru_RU.UTF-8
export LC_ALL=ru_RU.UTF-8

initdb -D "$PGDATA" --encoding=$PGENCODE --locale=$PGLOCALE
--lc-messages=$PGLOCALE --lc-monetary=$PGLOCALE --lc-numeric=$PGLOCALE
--lc-time=$PGLOCALE --no-locale --username=$PGUSERNAME

pg_ctl -D $PGDATA -l $PGDATA/server.log start

echo "listen_addresses = '*'" >> $PGDATA/postgresql.conf
echo "port = 9114" >> $PGDATA/postgresql.conf
cat > $PGDATA/pg_hba.conf << EOF
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             127.0.0.1/32            password
host    all             all             ::1/128                 password
local   all             all                                     peer
EOF

pg_ctl -D $PGDATA restart
```

```
pg_ctl -D $PGDATA restart
ожидание завершения работы сервера.... готово
сервер остановлен
ожидание запуска сервера....2025-04-21 00:38:07.076 MSK [47034] СООБЩЕНИЕ:  завершение вывода в stderr
2025-04-21 00:38:07.076 MSK [47034] ПОДСКАЗКА:  В дальнейшем протокол будет выводиться в "syslog".
 готово
сервер запущен
[postgres2@pg185 ~]$ psql -h 127.0.0.1 -U $PGUSERNAME -p 9114 postgres
psql (16.4)
Введите "help", чтобы получить справку.

postgres=# \q
```

Создадим символические ссылки для табличных пространств:

```
mkdir ~/idd21
mkdir ~/gzp28

ln -s /var/db/postgres1/idd21 /var/db/postgres2/idd21
ln -s /var/db/postgres1/gzp28 /var/db/postgres2/gzp28
```

Теперь восстановим БД из последнего дампа:

```
latest_backup=$(ls -t ~/backups/*.sql.gz | head -1)
unpacked_file="${latest_backup%.gz}"

gzip -d $latest_backup
psql -h localhost -p 9114 postgres -f $unpacked_file
```

```
[postgres2@pg185 ~]$ psql -h localhost -p 9114 postgres -f $unpacked_file
SET
SET
SET
CREATE ROLE
ALTER ROLE
CREATE ROLE
ALTER ROLE
CREATE ROLE
ALTER ROLE
```

Проверяем доступность данных:

```
[postgres2@pg185 ~]$ psql -h 127.0.0.1 -p 9114 postgres
psql (16.4)
Введите "help", чтобы получить справку.

postgres=# \c fakebrownroad
Вы подключены к базе данных "fakebrownroad" как пользователь "postgres2".
fakebrownroad=# select count(*) from table1;
 count
-------
   100
(1 строка)

fakebrownroad=#
```

Этап 3. Повреждение файлов БД

Чтобы симулировать сбой, удаляем с диска директорию WAL со всем содержимым:

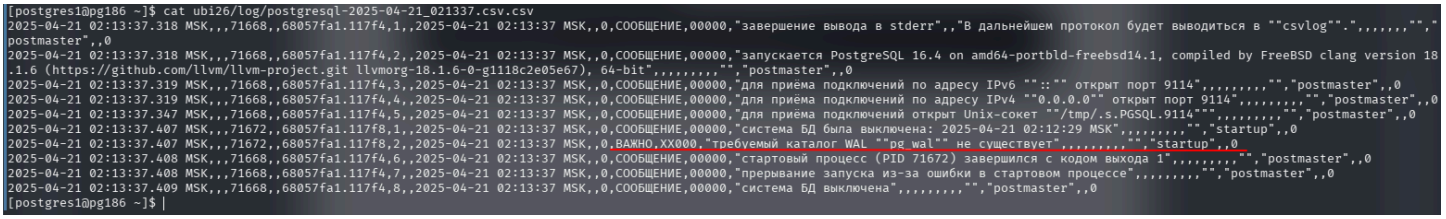
```
rm -rf $HOME/ubi26/pg_wal
```

Проверим работу СУБД и доступность данных:

```
[postgres1@pg186 ~]$ psql -h localhost -p 9114 postgres
psql (16.4)
Введите "help", чтобы получить справку.

postgres=# \q
[postgres1@pg186 ~]$
```

Перезапускаем СУБД и получаем ошибку:




```
ВАЖНО,XX000,"требуемый каталог WAL ""pg_wal"" не существует","startup",,0
```

Восстановление данных

Так как кластер не может запуститься без директории `pg_wal`, давайте выполним восстановление с последней резервной копии.

```
mkdir -p ~/restore/idd21 ~/restore/gzp28

latest_backup=$(ssh postgres2@pg185 "ls -t ~/backups/*.sql.gz | head -1")
yes | scp postgres2@pg185:$latest_backup ~/restore/

# Распаковка и замена путей табличных пространств
gzip -d ~/restore/pgdump-*.sql.gz
sed -i '' 's|/var/db/postgres1/idd21|/var/db/postgres1/restore/idd21|g'
~/restore/pgdump-*.sql
sed -i '' 's|/var/db/postgres1/gzp28|/var/db/postgres1/restore/gzp28|g'
~/restore/pgdump-*.sql
```

```
[postgres1@pg186 ~]$ mkdir ~/restore
[postgres1@pg186 ~]$ latest_backup=$(ssh postgres2@pg185 "ls -t ~/backups/*.sql.gz | head -1")
[postgres1@pg186 ~]$ scp postgres2@pg185:$latest_backup ~/restore/
pgdump-2025-04-21-02-00-00.sql.gz                                                    100% 2981     2.9MB/s   00:00
[postgres1@pg186 ~]$ gzip -d ~/restore/pgdump-*.sql.gz
[postgres1@pg186 ~]$ sed -i '' 's|/var/db/postgres1/idd21|/var/db/postgres1/restore/idd21|g' ~/restore/pgdump-*.sql
[postgres1@pg186 ~]$ sed -i '' 's|/var/db/postgres1/gzp28|/var/db/postgres1/restore/gzp28|g' ~/restore/pgdump-*.sql
[postgres1@pg186 ~]$ mkdir -p ~/restore/idd21 ~/restore/gzp28
[postgres1@pg186 ~]$
```

Инициализация и восстановление:

```
NEW_DATA_DIR="$HOME/ubi26_recovery"
mkdir -p $NEW_DATA_DIR

initdb -D $NEW_DATA_DIR --encoding=UTF8 --locale=ru_RU.UTF-8
--lc-messages=ru_RU.UTF-8 --lc-monetary=ru_RU.UTF-8
--lc-numeric=ru_RU.UTF-8 --lc-time=ru_RU.UTF-8 --no-locale
--username=postgres1
echo "listen_addresses = '*'" >> $NEW_DATA_DIR/postgresql.conf
echo "port = 9114" >> $NEW_DATA_DIR/postgresql.conf

# Запускаем PostgreSQL с новым каталогом
pg_ctl -D $NEW_DATA_DIR -l $NEW_DATA_DIR/server.log start

# Восстанавливаем данные (с учетом новых путей табличных пространств)
psql -h localhost -p 9114 postgres -f ~/restore/pgdump-*.sql
pg_ctl -D $NEW_DATA_DIR stop

# Перемещаем данные
rm -rf $HOME/ubi26
mv $NEW_DATA_DIR $HOME/ubi26

# Запускаем с оригинальным путем
pg_ctl -D $HOME/ubi26 start
```

```
[postgres1@pg186 ~]$ initdb -D $NEW_DATA_DIR --encoding=UTF8 --locale=ru_RU.UTF-8 --lc-messages=ru_RU.UTF-8 --lc-monetary=ru_RU.UTF-8
 --lc-numeric=ru_RU.UTF-8 --lc-time=ru_RU.UTF-8 --no-locale --username=postgres1
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres1".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован со следующими параметрами локали:
  провайдер:   libc
  LC_COLLATE:  C
  LC_CTYPE:    C
  LC_MESSAGES: ru_RU.UTF-8
  LC_MONETARY: ru_RU.UTF-8
  LC_NUMERIC:  ru_RU.UTF-8
  LC_TIME:     ru_RU.UTF-8
Выбрана конфигурация текстового поиска по умолчанию "english".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres1/ubi26_recovery ... ок
создание подкаталогов ... ок
выбирается реализация динамической разделяемой памяти ... posix
выбирается значение max_connections по умолчанию ... 100
выбирается значение shared_buffers по умолчанию ... 128MB
выбирается часовой пояс по умолчанию ... Europe/Moscow
создание конфигурационных файлов ... ок
выполняется подготовительный скрипт ... ок
выполняется заключительная инициализация ... ок
сохранение данных на диске ... ок

initdb: предупреждение: включение метода аутентификации "trust" для локальных подключений
initdb: подсказка: Другой метод можно выбрать, отредактировав pg_hba.conf или ещё раз запустив initdb с ключом -A, --auth-local или -
-auth-host.

Готово. Теперь вы можете запустить сервер баз данных:

    pg_ctl -D /var/db/postgres1/ubi26_recovery -l файл_журнала start

[postgres1@pg186 ~]$ echo "listen_addresses = '*'" >> $NEW_DATA_DIR/postgresql.conf
echo "port = 9114" >> $NEW_DATA_DIR/postgresql.conf
[postgres1@pg186 ~]$ pg_ctl -D $NEW_DATA_DIR -l $NEW_DATA_DIR/server.log start
```

```
[postgres1@pg186 ~]$ psql -h localhost -p 9114 postgres -f ~/restore/pgdump-*.sql
SET
SET
SET
CREATE ROLE
ALTER ROLE
CREATE ROLE
ALTER ROLE
psql:/var/db/postgres1/restore/pgdump-2025-04-21-02-00-00.sql:18: ОШИБКА:  роль "postgres1" уже существует
ALTER ROLE
CREATE TABLESPACE
GRANT
CREATE TABLESPACE
GRANT
Вы подключены к базе данных "template1" как пользователь "postgres1".
SET
SET
SET
SET
SET
 set_config
------------

(1 строка)

SET
SET
SET
SET
SET
SET
SET
SET
SET
 set_config
------------

(1 строка)
```

Перемещаем данные и запускаем с оригинальным путем:

```
[postgres1@pg186 ~]$ pg_ctl -D $NEW_DATA_DIR stop
ожидание завершения работы сервера..... готово
сервер остановлен
[postgres1@pg186 ~]$ rm -rf $HOME/ubi26/*
[postgres1@pg186 ~]$ mv $NEW_DATA_DIR $HOME/ubi26
[postgres1@pg186 ~]$ rmdir $NEW_DATA_DIR
rmdir: /var/db/postgres1/ubi26_recovery: No such file or directory
[postgres1@pg186 ~]$
[postgres1@pg186 ~]$ pg_ctl -D $HOME/ubi26 start
ожидание запуска сервера....2025-04-21 02:49:04.003 MSK [80444] СООБЩЕНИЕ:  завершение вывода в stderr
2025-04-21 02:49:04.003 MSK [80444] ПОДСКАЗКА:  В дальнейшем протокол будет выводиться в "syslog".
 готово
сервер запущен
[postgres1@pg186 ~]$
```

Проверим работу и доступность данных:

```
[postgres1@pg186 ~]$ psql -h 127.0.0.1 -p 9114 postgres
psql (16.4)
Введите "help", чтобы получить справку.

postgres=# \c fakebrownroad
Вы подключены к базе данных "fakebrownroad" как пользователь "postgres1".
fakebrownroad=# select count(*) from table1;
 count
-------
   100
(1 строка)

fakebrownroad=#
```

Данные успешно восстановились. Проверим, что пути дополнительных табличных пространств изменились корректно:

```
postgres=# \db+
                                                  Список табличных пространств
    Имя     | Владелец  |          Расположение          |     Права доступа     | Параметры | Размер | Описание
------------+-----------+--------------------------------+-----------------------+-----------+--------+----------
 pg_default | postgres1 |                                |                       |           | 30 MB  |
 pg_global  | postgres1 |                                |                       |           | 589 kB |
 ts1        | postgres1 | /var/db/postgres1/restore/idd21 | postgres1=C/postgres1+|           | 16 kB  |
            |           |                                | data_user=C/postgres1 |           |        |
 ts2        | postgres1 | /var/db/postgres1/restore/gzp28 | postgres1=C/postgres1+|           | 24 kB  |
            |           |                                | data_user=C/postgres1 |           |        |
(4 строки)
```

Этап 4. Логическое повреждение данных

Так как требуется восстановление с использованием архивных WAL-файлов, в начале нужно включить архивирование. Для этого нужно изменить конфигурацию:

```
# Настройка архивации WAL
psql -h localhost -p 9114 postgres <<EOF
ALTER SYSTEM SET wal_level = 'replica';
ALTER SYSTEM SET archive_mode = on;
ALTER SYSTEM SET archive_command = 'test ! -f $HOME/wal_archive/%f && cp
%p $HOME/wal_archive/%f';
EOF

# Создаём каталог для WAL-архивов
mkdir -p ~/wal_archive

pg_ctl -D $HOME/ubi26 restart
psql -h localhost -p 9114 postgres -c "SHOW archive_mode; SHOW
archive_command;"
```

```
[postgres1@pg186 ~]$ mkdir -p ~/wal_archive

pg_ctl -D $HOME/ubi26 restart
psql -h localhost -p 9114 postgres -c "SHOW archive_mode; SHOW archive_command;"
ожидание завершения работы сервера..... готово
сервер остановлен
ожидание запуска сервера....2025-04-28 01:55:42.327 MSK [74435] СООБЩЕНИЕ:  передача вывода в протокол процессу сбора протоколов
2025-04-28 01:55:42.327 MSK [74435] ПОДСКАЗКА:  В дальнейшем протоколы будут выводиться в каталог "log".
 готово
сервер запущен
 archive_mode
--------------
 on
(1 строка)

                               archive_command
-----------------------------------------------------------------------------
 test ! -f /var/db/postgres1/wal_archive/%f && cp %p /var/db/postgres1/wal_archive/%f
(1 строка)
```

Для PITR нам также понадобится базовый бэкап, сделанный перед изменениями:

```
cat >> $PGDATA/pg_hba.conf << EOF
host    replication  postgres1  ::1/128       trust
host    replication  postgres1  127.0.0.1/32  trust
EOF
pg_ctl -D $PGDATA restart
pg_basebackup -D $HOME/base_backup -h localhost -p 9114 -U postgres1 -P
-Ft -Xs -z
```

```
pg_ctl -D $PGDATA restart
pg_basebackup -D $HOME/base_backup -h localhost -p 9114 -U postgres1 -P -Ft -Xs -z

ожидание завершения работы сервера..... готово
сервер остановлен
ожидание запуска сервера....2025-04-28 01:56:58.453 MSK [74557] СООБЩЕНИЕ:  передача вывода в протокол процессу сбора протоколов
2025-04-28 01:56:58.453 MSK [74557] ПОДСКАЗКА:  В дальнейшем протоколы будут выводиться в каталог "log".
 готово
сервер запущен
31275/31275 КБ (100%), табличное пространство 3/3
[postgres1@pg186 ~]$
```

В каждую таблицу базы добавим 2-3 новые строки и зафиксируем результат:

```
psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "
INSERT INTO table1 (name) SELECT 'До сбоя ' || g FROM
generate_series(101, 103) g;
INSERT INTO table2 (value) SELECT g*10 FROM generate_series(101, 103) g;
INSERT INTO table3 (info) SELECT 'До сбоя ' || g FROM
generate_series(101, 103) g;"
```

```
[postgres1@pg186 ~]$ psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "
INSERT INTO table1 (name) SELECT 'До сбоя ' || g FROM generate_series(101, 103) g;
INSERT INTO table2 (value) SELECT g * 10 FROM generate_series(101, 103) g;
INSERT INTO table3 (info) SELECT 'До сбоя ' || g FROM generate_series(101, 103) g;
"
INSERT 0 3
INSERT 0 3
INSERT 0 3
[postgres1@pg186 ~]$ psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "select count(*) from table1;"
 count
-------
   103
(1 строка)

[postgres1@pg186 ~]$
```

Фиксируем время сбоя и симулируем ошибку:

```
SCHEMA_TIME=$(psql -h localhost -p 9114 -U postgres1 -d fakebrownroad -t
-c "SELECT now();")
echo "Цель восстановления: $SCHEMA_TIME"

psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "DELETE FROM
table1 WHERE id % 2 = 1;"
```

```
[postgres1@pg186 ~]$ SCHEMA_TIME=$(psql -h localhost -p 9114 -U postgres1 -d fakebrownroad -t -c "SELECT now();")
echo "Цель восстановления: $SCHEMA_TIME"
Цель восстановления:  2025-04-28 02:03:28.611482+03
[postgres1@pg186 ~]$ psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "DELETE FROM table1 WHERE id % 2 = 1;"
DELETE 52
[postgres1@pg186 ~]$
```

Демонстрация результата:

```
psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "SELECT
count(*) FROM table1; SELECT count(*) FROM table2;"
```

```
[postgres1@pg186 ~]$ psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "SELECT count(*) FRO
M table1; SELECT count(*) FROM table2;"
 count
-------
    51
(1 строка)

 count
-------
   103
(1 строка)

[postgres1@pg186 ~]$
```

Восстановление данных

Остановим сервер БД:

```
pg_ctl -D $HOME/ubi26 stop
```

Очищаем каталог данных и используем базовый бэкап:

```
rm -rf $HOME/ubi26/*
tar -xvf $HOME/base_backup/base.tar.gz -C $HOME/ubi26/
```

```
[postgres1@pg186 ~]$
rm -rf $HOME/ubi26/*
tar -xvf $HOME/base_backup/base.tar.gz -C $HOME/ubi26/

x backup_label
x tablespace_map
x current_logfiles
x pg_commit_ts/
x pg_replslot/
x pg_stat/
x pg_dynshmem/
x pg_multixact/
x pg_multixact/members/
x pg_multixact/members/0000
x pg_multixact/offsets/
x pg_multixact/offsets/0000
x pg_xact/
x pg_xact/0000
x pg_wal/
x ./pg_wal/archive_status/
x postgresql.conf
x base/
x base/5/
```

Попробуем восстановиться через `restore_command`.

```
cat >> $HOME/ubi26/postgresql.conf <<EOF
restore_command = 'cp $HOME/wal_archive/%f %p'
recovery_target_time = '$SCHEMA_TIME'
EOF
# Создаем сигнальный файл для запуска в режиме восстановления
touch $HOME/ubi26/recovery.signal
# Запуск восстановления
pg_ctl -D $HOME/ubi26 -l $HOME/ubi26/recovery.log start
# Мониторинг процесса
tail -f $HOME/ubi26/recovery.log
```

```
[postgres1@pg186 ~]$ cat ubi26/log/postgresql-2025-04-28_020727.csv
2025-04-28 02:07:27.165 MSK,,,75618,,680eb8af.12762,1,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"завершение вывода в stderr",,"В дальнейшем протокол будет выводиться в ""csvlog"".",,,,,,,"","","postmaster",,0
2025-04-28 02:07:27.165 MSK,,,75618,,680eb8af.12762,2,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"запускается PostgreSQL 16.4 on amd64-portbld-freebsd14.1, compiled by FreeBSD clang version 18.1.6 (https://github.com/llvm/llvm-project.git llvmorg-18.1.6-0-g1118c2e05e67), 64-bit",,,,,,,,,"","postmaster",,0
2025-04-28 02:07:27.165 MSK,,,75618,,680eb8af.12762,3,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"для приёма подключений по адресу IPv6 ""::"" открыт порт 9114",,,,,,,,,"","postmaster",,0
2025-04-28 02:07:27.165 MSK,,,75618,,680eb8af.12762,4,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"для приёма подключений по адресу IPv4 ""0.0.0.0"" открыт порт 9114",,,,,,,,,"","postmaster",,0
2025-04-28 02:07:27.181 MSK,,,75618,,680eb8af.12762,5,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"для приёма подключений открыт Unix-сокет ""/tmp/.s.PGSQL.9114""",,,,,,,,,"","postmaster",,0
2025-04-28 02:07:27.253 MSK,,,75622,,680eb8af.12766,1,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"работа системы БД была прервана; последний момент работы: 2025-04-28 02:02:38 MSK",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.350 MSK,,,75622,,680eb8af.12766,2,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"начинается восстановление точки во времени до 2025-04-28 02:03:28.611482+03",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.363 MSK,,,75622,,680eb8af.12766,3,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"начинается восстановление копии с LSN redo 0/3000028, LSN контрольной точки 0/3000060, на линии времени 1",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.366 MSK,,,75622,,680eb8af.12766,4,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"файл журнала ""000000010000000000000003"" восстановлен из архива",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.498 MSK,,,75622,,680eb8af.12766,5,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"запись REDO начинается со смещения 0/3000028",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.533 MSK,,,75622,,680eb8af.12766,6,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"файл журнала ""000000010000000000000004"" восстановлен из архива",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.625 MSK,,,75622,,680eb8af.12766,7,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"завершено восстановление копии с LSN redo 0/3000028 и конечным LSN 0/3000100",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.625 MSK,,,75622,,680eb8af.12766,8,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"согласованное состояние восстановления достигнуто в позиции 0/3000100",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.625 MSK,,,75618,,680eb8af.12762,6,,2025-04-28 02:07:27 MSK,,0,СООБЩЕНИЕ,00000,"система БД готова принимать подключения в режиме ""только чтение""",,,,,,,,,"","postmaster",,0
2025-04-28 02:07:27.670 MSK,,,75622,,680eb8af.12766,9,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"восстановление останавливается перед фиксированием транзакции 761, время 2025-04-28 02:03:31.370453+03",,,,,,,,,"","startup",,0
2025-04-28 02:07:27.670 MSK,,,75622,,680eb8af.12766,10,,2025-04-28 02:07:27 MSK,1/0,0,СООБЩЕНИЕ,00000,"остановка в конце восстановления",,"Выполните pg_wal_replay_resume() для повышения.",,,,,,"","startup",,0
[postgres1@pg186 ~]$
```

Судя по логам, данные успешно восстановились. Проверим:

```
[postgres1@pg186 ~]$ psql -h localhost -p 9114 -U data_user -d fakebrownroad -t -c "SELECT count(*) FROM table1;"
   103
```

Таблица вернулась в состояние до удаления половины элементов.

После этого не забудем прописать функцию `pg_wal_replay_resume()` чтобы завершить восстановление. После этого можно спокойно продолжать работать с БД. Если не прописать эту функцию, тогда у нас восстановление не завершится и при перезапуске без файла `recovery.signal` у нас все откатится к последней версии как было после `DELETE`.

Подтверждаем завершение восстановления:

```
psql -h localhost -p 9114 postgres -c "SELECT pg_wal_replay_resume();"
```

Удаляем сигнальный файл:

```
rm $HOME/ubi26/recovery.signal
```

Убираем параметры восстановления

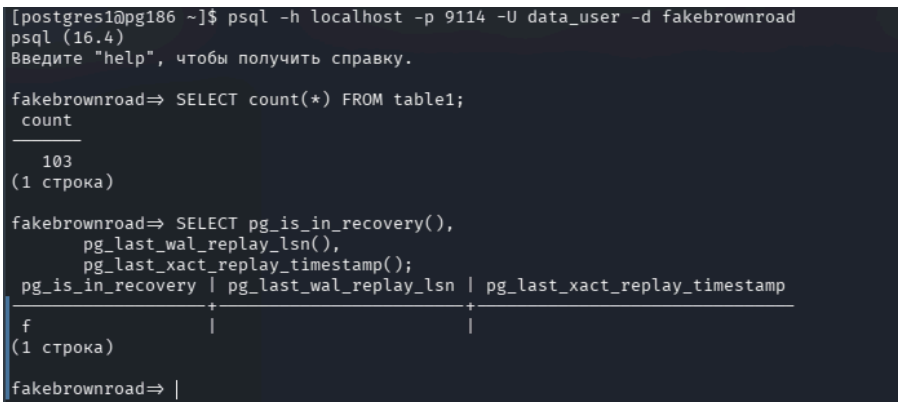
```
sed -i '' '/^restore_command/d' $HOME/ubi26/postgresql.conf
sed -i '' '/^recovery_target_time/d' $HOME/ubi26/postgresql.conf
```

Проверка результатов вне режима восстановления:

```
psql -h localhost -p 9114 postgres -c "
SELECT pg_is_in_recovery(),
       pg_last_wal_replay_lsn(),
       pg_last_xact_replay_timestamp();"

# Проверка восстановленных данных
psql -h localhost -p 9114 -U data_user -d fakebrownroad -c "SELECT
count(*) FROM table1;"
```

После успешного восстановления:



Исходный код

https://github.com/maxbarsukov/itmo/tree/master/6%20pcxд/лабораторные/lab3

Вывод

В ходе выполнения лабораторной работы я познакомился с резервным копированием кластера PostgreSQL и на практике изучил способ непрерывного PITR бекапа базы данных, настроил и применил резервное копирование и восстановление при различных сбоях: полной потере основного узла, повреждении файлов БД или логическом повреждении данных.