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 3](#_4wv7arx2x4xz)

[Выполнение 5](#_z5ouzacohqk7)

[Этап 1. Резервное копирование 5](#_2gkk9zdm7y4j)

[Расчет объема резервных копий 6](#_nfrs2am4jmyf)

[Анализ 7](#_sydfjd893kaq)

[Этап 2. Потеря основного узла 8](#_wcl09aujm2vt)

[Восстановление данных 8](#_hqdgvtcijmlh)

[Этап 3. Повреждение файлов БД 10](#_ydelrnterpup)

[Восстановление данных 10](#_7n2dicx3yllg)

[Этап 4. Логическое повреждение данных 13](#_tufilqc4wosw)

[Восстановление данных 15](#_c62l1rjntc17)

[Вывод](#_9nxi2mdwsuhw) 18

# **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.



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**:



Резервный узел **pg185**:



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.



**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: МБ.

Размер сжатого дампа на день t: МБ.

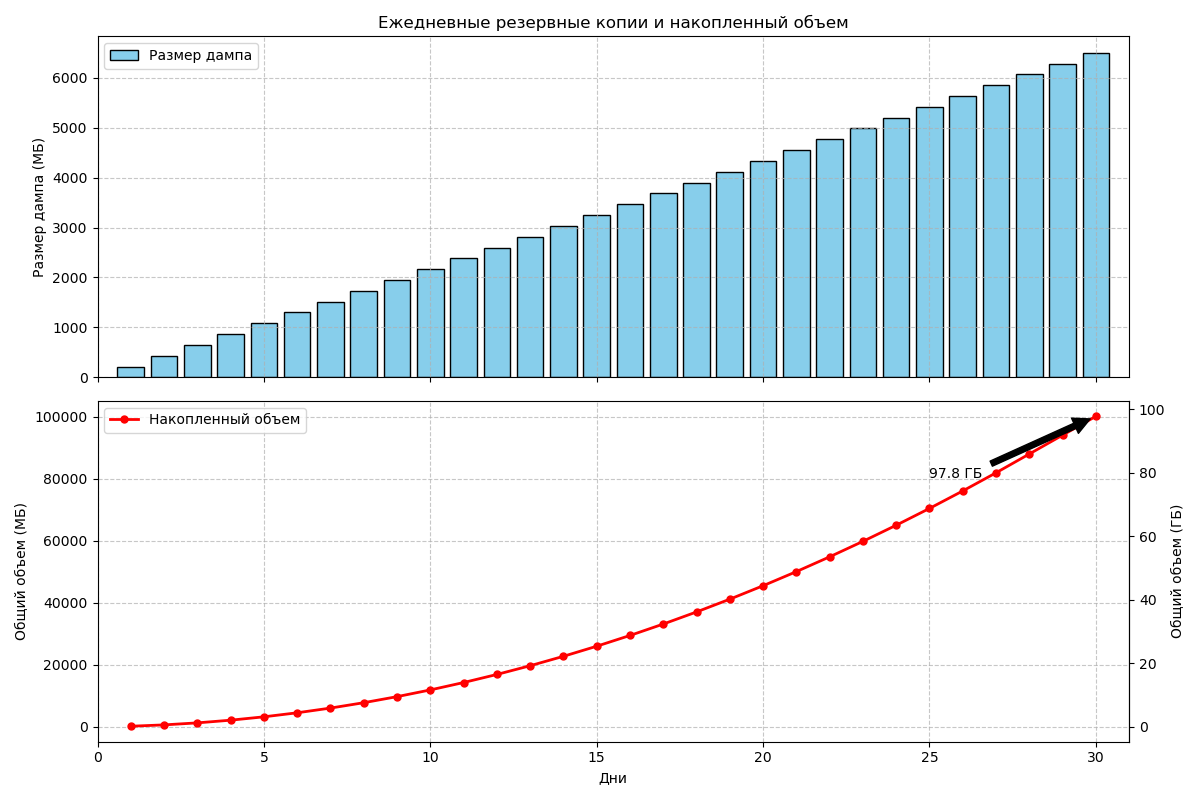
Общий объем на 30-й день:

### **Анализ**

Дамп на 30-й день: 6.5 ГБ.

Средний размер дампа: 3.28 ГБ/день.

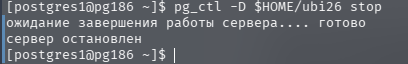
Пиковый объём: 6.5 ГБ (последний день).



## **Этап 2. Потеря основного узла**

Имитируем полную недоступность основного узла остановкой PostgreSQL:

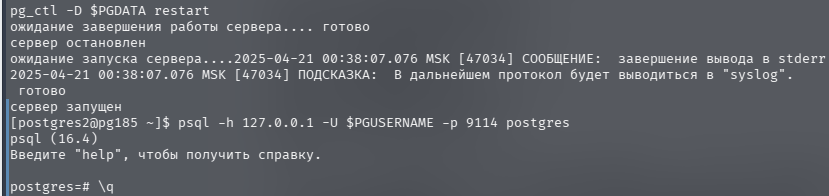
| pg\_ctl -D $HOME/ubi26 stop |
| --- |



### **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 |
| --- |

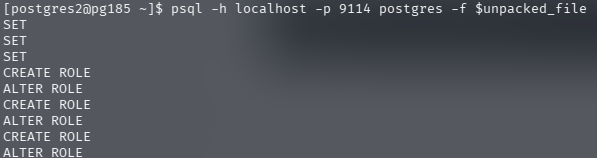


Создадим директории для табличных пространств:

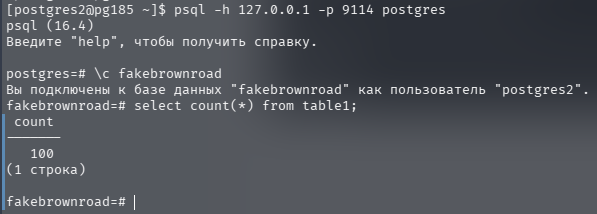
| mkdir ~/idd21  mkdir ~/gzp28 |
| --- |

Теперь восстановим БД из последнего дампа:

| latest\_backup=$(ls -t ~/backups/\*.sql.gz | head -1)  unpacked\_file="${latest\_backup%.gz}"  gzip -d $latest\_backup  sed -i '' 's|/var/db/postgres1/idd21|/var/db/postgres2/idd21|g' $unpacked\_file  sed -i '' 's|/var/db/postgres1/gzp28|/var/db/postgres2/gzp28|g' $unpacked\_file  psql -h localhost -p 9114 -U postgres1 postgres -f $unpacked\_file |
| --- |



Проверяем доступность данных:

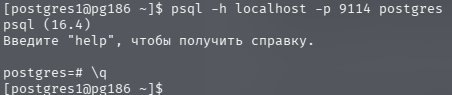
****

## **Этап 3. Повреждение файлов БД**

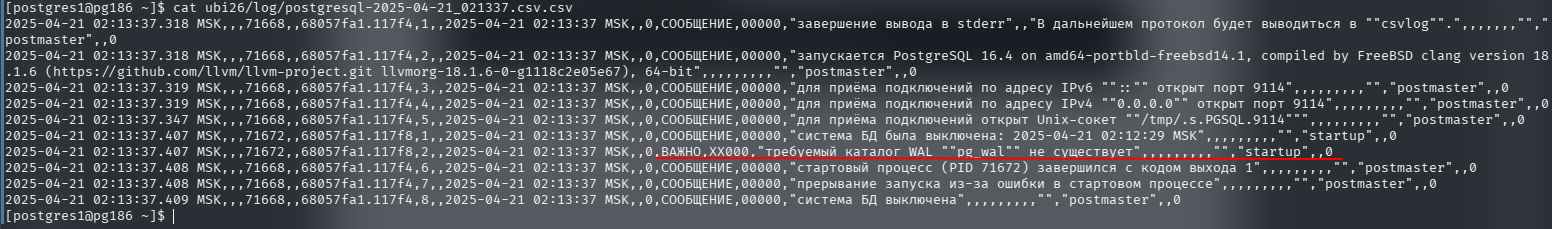
Чтобы симулировать сбой, удаляем с диска директорию WAL со всем содержимым:

| rm -rf $HOME/ubi26/pg\_wal |
| --- |

Проверим работу СУБД и доступность данных:



Перезапускаем СУБД и получаем ошибку:

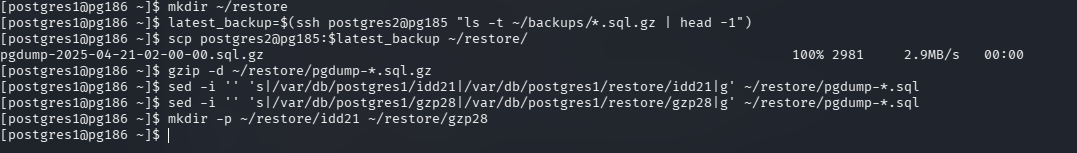


ВАЖНО,XX000,"требуемый каталог WAL ""pg\_wal"" не существует","startup",,0

### **Восстановление данных**

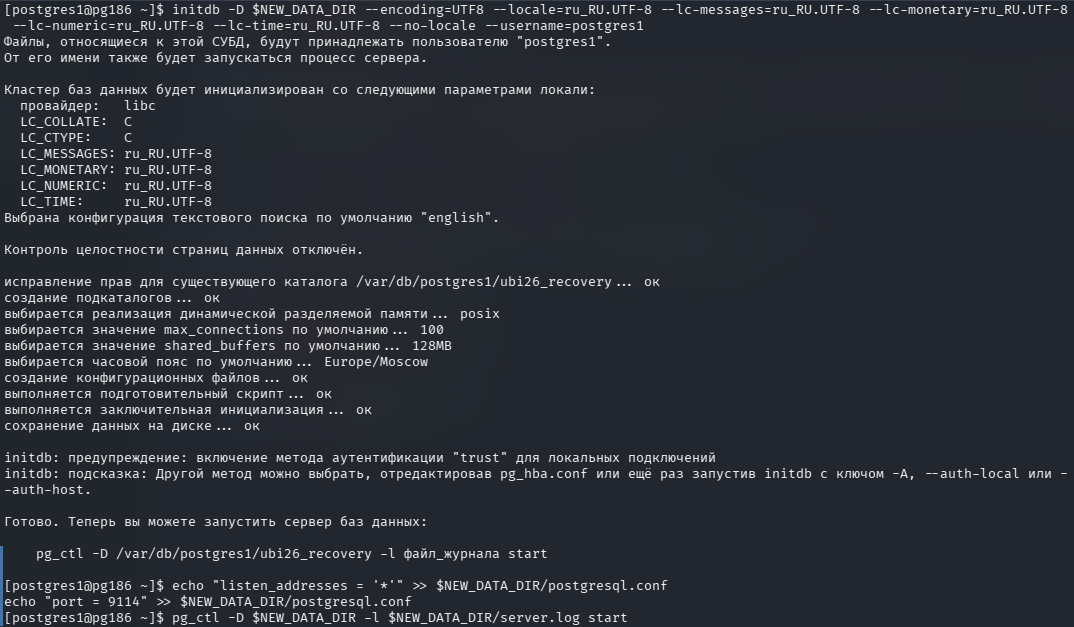
Так как кластер не может запуститься без директории pg\_wal, давайте выполним восстановление с последней резервной копии.

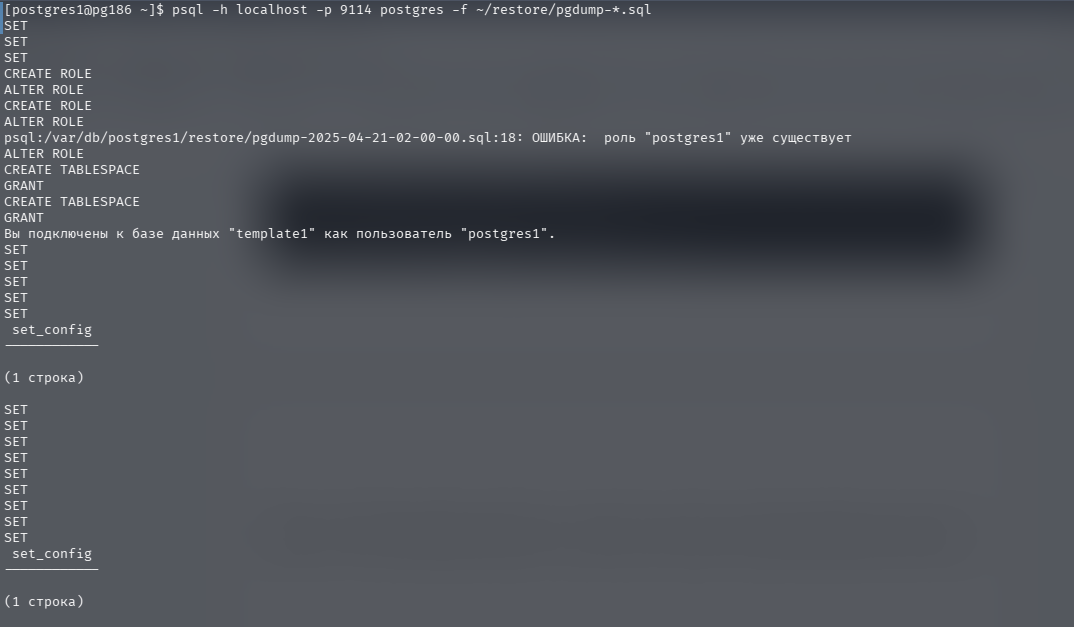
| mkdir -p ~/restore/idd21 ~/restore/gzp28  latest\_backup=$(ssh postgres2@pg185 "ls -t ~/backups/\*.sql.gz | head -1")  yes | scp postgres2@pg185:$latest\_backup ~/restore/  # Распаковка и замена путей табличных пространств  gzip -d ~/restore/pgdump-\*.sql.gz  sed -i '' 's|/var/db/postgres1/idd21|/var/db/postgres1/restore/idd21|g' ~/restore/pgdump-\*.sql  sed -i '' 's|/var/db/postgres1/gzp28|/var/db/postgres1/restore/gzp28|g' ~/restore/pgdump-\*.sql |
| --- |



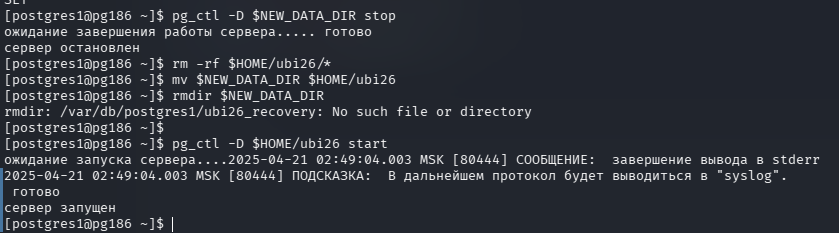
**Инициализация и восстановление:**

| 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 |
| --- |

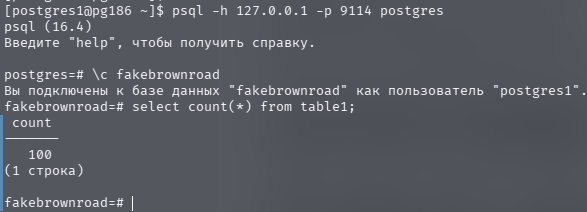




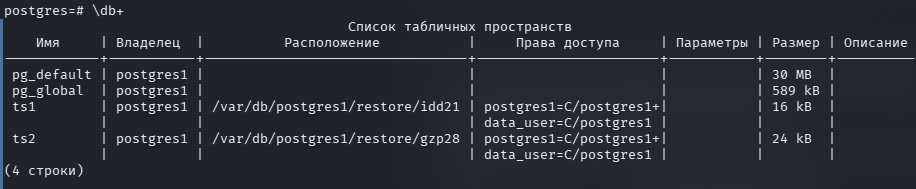
Перемещаем данные и запускаем с оригинальным путем:



Проверим работу и доступность данных:



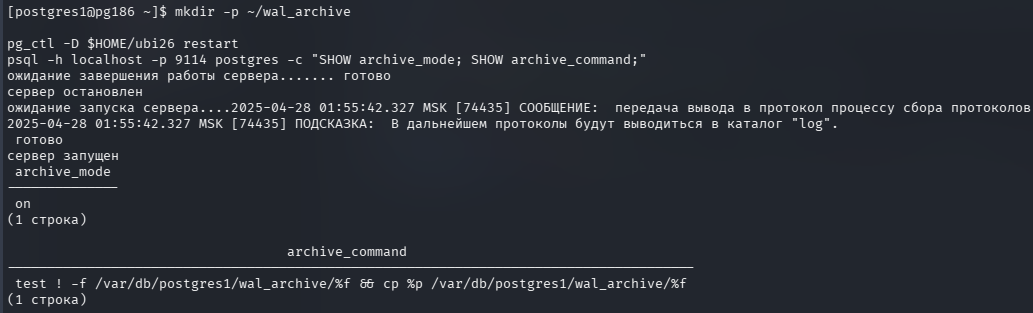
Данные успешно восстановились. Проверим, что пути дополнительных табличных пространств изменились корректно:



## **Этап 4. Логическое повреждение данных**

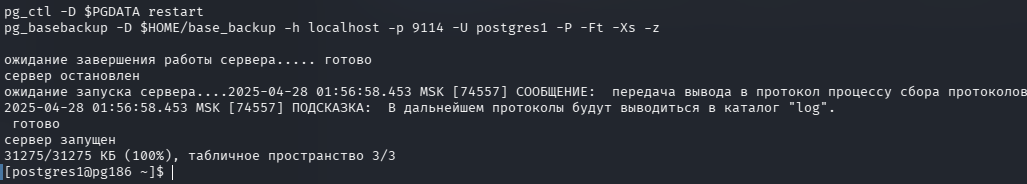
Так как требуется восстановление с использованием архивных WAL-файлов, в начале нужно включить архивирование. Для этого нужно изменить конфигурацию:

| # 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;" |
| --- |



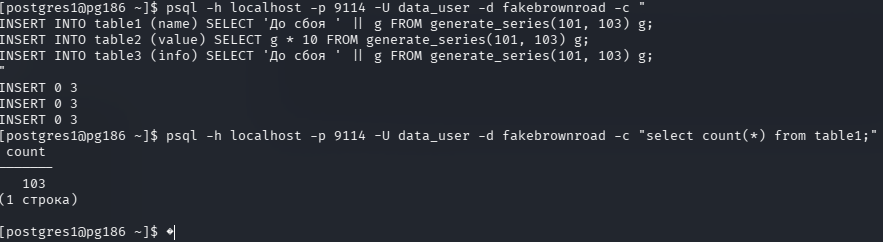
Для PITR нам также понадобится базовый бэкап, сделанный перед изменениями:

| 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 |
| --- |



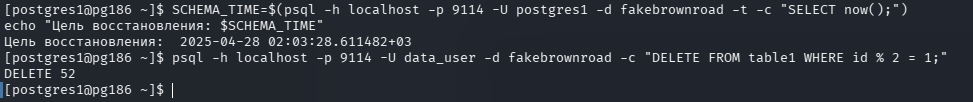
В каждую таблицу базы добавим 2-3 новые строки и зафиксируем результат:

| 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;" |
| --- |



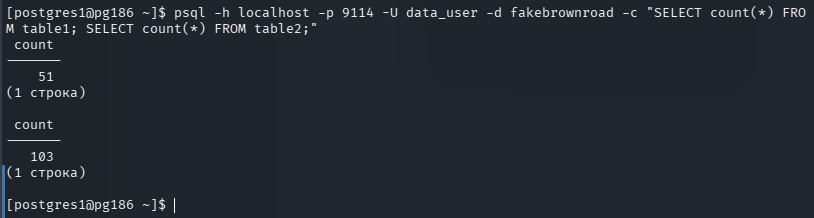
Фиксируем время сбоя и симулируем ошибку:

| 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;" |
| --- |



Демонстрация результата:

| psql -h localhost -p 9114 -U data\_user -d fakebrownroad -c "SELECT count(\*) FROM table1; SELECT count(\*) FROM table2;" |
| --- |



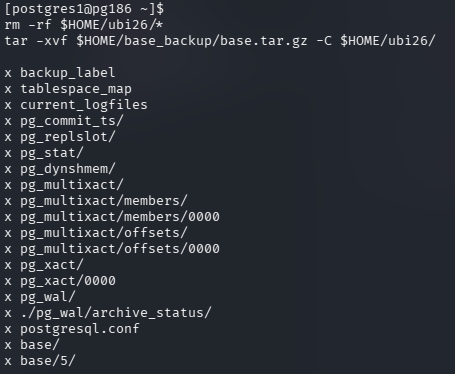
### **Восстановление данных**

Остановим сервер БД:

| pg\_ctl -D $HOME/ubi26 stop |
| --- |

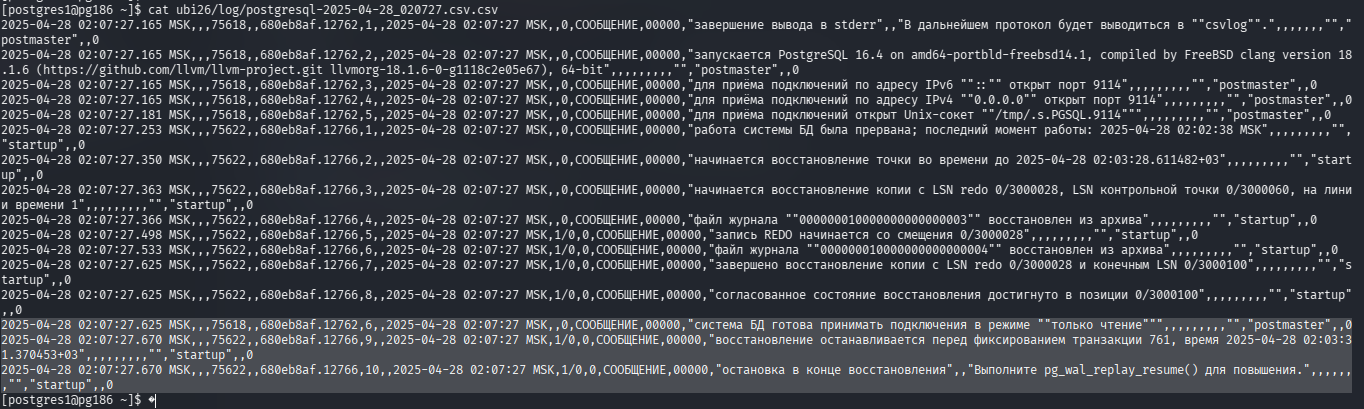
Очищаем каталог данных и используем базовый бэкап:

| rm -rf $HOME/ubi26/\*  tar -xvf $HOME/base\_backup/base.tar.gz -C $HOME/ubi26/ |
| --- |



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 |
| --- |



Судя по логам, данные успешно восстановились. Проверим:



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 |
| --- |

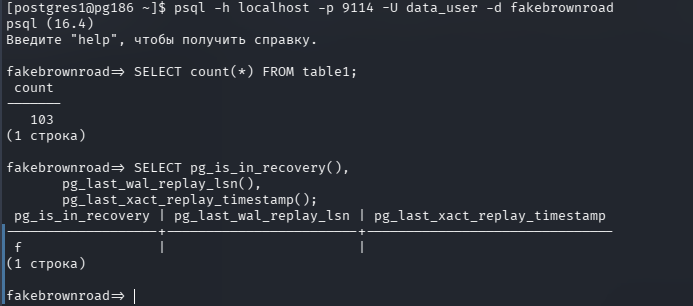
Убираем параметры восстановления

| 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%20рсхд/лабораторные/lab3](https://github.com/maxbarsukov/itmo/tree/master/6%20%D1%80%D1%81%D1%85%D0%B4/%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D1%8B%D0%B5/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.