

**Министерство науки и высшего образования Российской Федерации**  
Федеральное государственное автономное образовательное учреждение высшего  
образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Дисциплина: Администрирование систем управления базами данных**

**Лабораторная работа №3**  
Вариант 3800

Выполнил:  
Серебренникова В. В.

Группа:  
Р33202

Проверил:  
Николаев В. В.

Санкт-Петербург  
2024

## Оглавление

Оглавление.....	2
Описание задания .....	3
Отчет .....	5
Этап 1.....	5
Этап 2.....	10
Этап 3.....	14
Этап 4.....	19
Выводы.....	28

## Описание задания

### Вариант 3800

Номер основного узла: pg189, пользователь: postgres0.

Номер резервного узла: pg169 пользователь: postgres6.

Пароли выданы преподавателем на почту.

### 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 выдаёт преподаватель. В сценариях восстановления необходимо использовать копию данных, полученную на первом этапе данной лабораторной работы.

Способ подключения к узлу из сети Интернет через helios:

ssh -J sXXXXXX@helios.cs.ifmo.ru:2222 postgresY@pgZZZ

Способ подключения к узлу из сети факультета:

ssh

postgresY@pgZZZ

Этап	1.	Резервное	копирование
------	----	-----------	-------------

- Настроить резервное копирование с основного узла на резервный следующим образом:
  - Периодические обособленные (standalone) полные копии.
  - Полное резервное копирование (pg\_basebackup) по расписанию (cron) два раза в сутки.
  - Необходимые файлы WAL должны быть в составе полной копии, отдельно их не архивировать.
  - Срок хранения копий на основной системе - 1 неделя, на резервной - 1 месяц.
  - 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: 400МБ.
  - Средний объем измененных данных за сутки: 250МБ.
- 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.

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 конфигурационных файлов СУБД 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.

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:
  - Удалить каждую вторую строку в любой таблице (DELETE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

# Отчет

## Этап 1

Настроим резервное копирование с основного узла (postgres0@pg189) на резервный (postgres6@pg169). Для начала попробуем подключиться из основного узла к резервному по выданным преподавателем номеру пользователя, паролю и узлу:

```
[postgres0@pg189 ~/dyu42]$ ssh postgres6@pg169
The authenticity of host 'pg169.cs.ifmo.ru (192.168.11.169)' can't be established.
ECDSA key fingerprint is SHA256:DtU400PFfPRML9bA2qG0yYt41kcMhbiIhSyPPXWgZJk.
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pg169.cs.ifmo.ru' (ECDSA) to the list of known hosts.
Password for postgres6@pg169.cs.ifmo.ru:
[postgres6@pg169 ~]$
```

Рис. 1 - Подключение к резервному узлу

Копирование при помощи pg\_basebackup - физическое копирование. Для его организации необходимо изменить следующие параметры (выполним на основном узле, рис. 2, 3):

- archive\_mode = on
- wal\_level = replica

```
# - Settings -

wal_level = replica                # minimal, replica, or logical
                                   # (change requires restart)
fsync = on                         # flush data to disk for crash safety
                                   # (turning this off can cause
                                   # unrecoverable data corruption)
```

Рис. 2 - Изменение wal\_level

```
# - Archiving -

archive_mode = on                  # enables archiving; off, on, or always
                                   # (change requires restart)
archive_command = 'cp %p $HOME/dyu42/pg_wal' # command to use to archive a logfile segment
                                   # placeholders: %p = path of file to archive
                                   # %f = file name only
                                   # e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
archive_timeout = 0                # force a logfile segment switch after this
                                   # number of seconds; 0 disables
```

Рис. 3 - Изменение archive\_mode

Так как значение “all” в конфигурационном файле pg\_hba.conf в графе “база данных” не касается репликации, разрешим репликацию по хосту.

host	replication	all	127.0.0.1/32	trust
host	replication	all	:::1/128	trust

Рис. 4 - Разрешение репликации

Создадим кластер баз данных аналогично тому, как мы делали это в лабораторной работе №2.

Кластер понадобится нам, когда мы займемся восстановлением.

```
[postgres6@pg169 ~]$ initdb --encoding=UTF8 --locale=en_GB.UTF-8 -D $HOME/dyu42
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres6".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "en_GB.UTF-8".
Выбрана конфигурация текстового поиска по умолчанию "english".
!
Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres6/dyu42... 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... W-SU
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
Другой метод можно выбрать, отредактировав pg_hba.conf или используя ключи -A,
--auth-local или --auth-host при следующем выполнении initdb.
!
Готово. Теперь вы можете запустить сервер баз данных:

pg_ctl -D /var/db/postgres6/dyu42 -l файл_журнала start
```

Рис. 5 - Инициализация кластера баз данных

Чтобы не заморачиваться изменением конфигурационных файлов нового кластера для переноса параметров кластера и разрешений пользователей, удалим автоматически созданные файлы с резервного узла и скопируем их туда с основного.

```

[postgres6@pg169 ~/dyu42]$ ls
base                pg_ident.conf      pg_serial           pg_tblspc           postgresql.auto.conf
global              pg_logical          pg_snapshots        pg_twophase          postgresql.conf
pg_commit_ts        pg_multixact        pg_stat             PG_VERSION
pg_dynshmem         pg_notify          pg_stat_tmp         pg_wal
pg_hba.conf         pg_replslot        pg_subtrans         pg_xact
[postgres6@pg169 ~/dyu42]$ rm pg_hba.conf postgresql.conf
[postgres6@pg169 ~/dyu42]$ ls
base                pg_ident.conf      pg_replslot         pg_stat_tmp         PG_VERSION
global              pg_logical          pg_serial           pg_subtrans         pg_wal
pg_commit_ts        pg_multixact        pg_snapshots        pg_tblspc           pg_xact
pg_dynshmem         pg_notify          pg_stat             pg_twophase         postgresql.auto.conf
[postgres6@pg169 ~/dyu42]$ exit
выход
Connection to pg169.cs.ifmo.ru closed.
[postgres0@pg189 ~/dyu42]$ scp pg_hba.conf postgresql.conf postgres6@pg169.cs.ifmo.ru:/var/db/postgres6/dyu42
Password for postgres6@pg169.cs.ifmo.ru:
pg_hba.conf                                100% 4961  166.2KB/s   00:00
postgresql.conf                           100%  28KB  33.5MB/s   00:00
[postgres0@pg189 ~/dyu42]$ ssh postgres6@pg169
Password for postgres6@pg169.cs.ifmo.ru:
Last login: Tue May 28 16:59:32 2024 from 192.168.11.189
[postgres6@pg169 ~]$ ls dyu42
base                pg_ident.conf      pg_serial           pg_tblspc           postgresql.auto.conf
global              pg_logical          pg_snapshots        pg_twophase          postgresql.conf
pg_commit_ts        pg_multixact        pg_stat             PG_VERSION
pg_dynshmem         pg_notify          pg_stat_tmp         pg_wal
pg_hba.conf         pg_replslot        pg_subtrans         pg_xact
[postgres6@pg169 ~]$

```

Рис. 6 - Копирование конфигурационных файлов

Создадим скрипт .sh, в котором будет описан процесс резервного копирования (WAL-файлы переносим вместе с бэкап-файлами, как указано в варианте).

```

BACKUP_SERVER_PATH=postgres6@pg169:/var/db/postgres6/backups
BACKUP_NAME=backup_$(date +"%Y-%m-%d-%H-%M")

pg_basebackup -p 9812 -h localhost --username=postgres0 -D $HOME/basebackups/$BACKUP_NAME -Ft -z -Xs

scp -r $HOME/basebackups/$BACKUP_NAME $BACKUP_SERVER_PATH

```

Рис. 7 - Содержимое pg\_backup.sh

Запустим скрипт. Как видно, резервная копия успешно перенеслась на резервный узел postgres6@pg169.

```

[postgres0@pg189 ~]$ bash pg_backup.sh
Password for postgres6@pg169.cs.ifmo.ru:
backup_manifest                                100% 184KB 104.0MB/s   00:00
16384.tar.gz                                  100%  144    1.0MB/s   00:00
base.tar.gz                                   100% 4521KB 154.7MB/s   00:00
pg_wal.tar.gz                                100%  17KB  43.9MB/s   00:00
16385.tar.gz                                  100%  142    1.0MB/s   00:00
[postgres0@pg189 ~]$ ssh

```

Рис. 8 - Процесс резервного копирования

```
[postgres6@pg169 ~/backups]$ ls backup_2024-05-30-20-59
16384.tar.gz  16385.tar.gz  backup_manifest base.tar.gz    pg_wal.tar.gz
[postgres6@pg169 ~/backups]$
```

Рис. 9 - Скопированные файлы на резервном узле

Настроим срок хранения копий на основном узле и сделаем его равным одной неделе (7 дней). Для этого создадим скрипт `backup_cleanup.sh`. В нем опишем команды для поиска и удаления файлов и директорий, используемых для копирования, возрастом более одной недели.

```
[postgres0@pg189 ~]$ vi backup_cleanup.sh
find $HOME/basebackups -type d -name "backup_*" -mtime +7 -exec rm -r {} +
find $HOME/dyu42/pg_wal -type f -name "*" -mtime +7 -exec rm -r {} +
~
```

Рис. 10 - Скрипт для удаления файлов на основном узле

Так как копии создаются дважды в день, скрипт для удаления копий так же будет запускаться дважды в день.

```
backup_cleanup.sh: 2 строк, 119 символов
[postgres0@pg189 ~]$ crontab -e
30 * * * * /var/db/postgres0/cleanup.sh
00 01,13 * * * /var/db/postgres0/pg_backup.sh
00 02,14 * * * /var/db/postgres0/backup_cleanup.sh
~
```

Рис. 11 - Расписание удаления файлов на основном узле

Настроим срок хранения копий на основном узле и сделаем его равным одному месяцу. Так как месяц - понятие растяжимое, я определила возраст записей для удаления не менее 30 дней.

```
find $HOME/backups -type d -name "backup_*" -mtime +30 -exec rm -r {} +
```

Рис. 12 - Скрипт для удаления файлов на резервном узле.

Так как копии создаются дважды в день, скрипт для удаления копий так же будет запускаться дважды в день.

```
[postgres6@pg169 ~]$ crontab -e
00 02,14 * * * /var/db/postgres6/backup_cleanup.sh
~
```

Рис. 13 - Расписание удаления файлов на резервном узле

Теперь займемся подсчетом. Посмотрим вес одного резервного копирования (архивы + WAL-файлы).



```
[postgres0@pg189 ~/basebackups]$ du -sh backup_2024-05-28-17
4,4M    backup_2024-05-28-17
[postgres0@pg189 ~/basebackups]$
```

Рис. 14 - Вес одного бэкапа

Вес одного резервного копирования с учетом всех архивов и WAL-файлов - 4,4 МБ. Учитывая, что удаляются копии на с резервного узла каждые 30 дней, а создаются дважды в день, максимальное единовременное количество копий на резервном узле равно 60 копиям. Получается, что максимальный вес директории с копиями будет равен  $4,4 * 60 = 264$  МБ.

Однако, учитывая, что каждый день в БД добавляется 400 МБ, нам нужно посчитать размер иначе. Посмотрим, сколько весит оригинальная директория с кластером, чтобы узнать, как сильно сжимаются файлы при копировании.

```
[postgres0@pg189 ~]$ du -sh dyu42
10M    dyu42
[postgres0@pg189 ~]$
```

Рис. 15 - Вес директории dyu42

Директория dyu42 весит 10 МБ, что значит, что вес директории при копировании уменьшается на 56%.

Посчитаем общий размер 60 копий без учета WAL-файлов по истечении 30 дней резервного копирования. Теперь подсчитаем размер WAL-файлов.

```
logfile pg_logical
[postgres0@pg189 ~/dyu42]$ du -sh pg_wal
157K    pg_wal
[postgres0@pg189 ~/dyu42]$
```

Рис. 16 - Размер pg\_wal

Оригинально директория занимает совсем мало места, поэтому я не буду учитывать эти 157 КБ при расчете.

Данные хранятся сегментами по 16 МБ, и незаполненные сегменты тоже занимают место. На изменения 250 МБ будет выделено 16 сегментов, что равно 256 МБ. За месяц это число дорастет до 7,5 ГБ.

День	Вес dyu42	Вес копии	Вес WAL
1	10,00	4,40	0
2	410,00	180,40	256
3	810,00	356,40	512
4	1210,00	532,40	768
5	1610,00	708,40	1024
6	2010,00	884,40	1280
7	2410,00	1060,40	1536
8	2810,00	1236,40	1792
9	3210,00	1412,40	2048
10	3610,00	1588,40	2304
11	4010,00	1764,40	2560
12	4410,00	1940,40	2816
13	4810,00	2116,40	3072
14	5210,00	2292,40	3328
15	5610,00	2468,40	3584
16	6010,00	2644,40	3840
17	6410,00	2820,40	4096
18	6810,00	2996,40	4352
19	7210,00	3172,40	4608
20	7610,00	3348,40	4864
21	8010,00	3524,40	5120
22	8410,00	3700,40	5376
23	8810,00	3876,40	5632
24	9210,00	4052,40	5888
25	9610,00	4228,40	6144
26	10010,00	4404,40	6400
27	10410,00	4580,40	6656
28	10810,00	4756,40	6912
29	11210,00	4932,40	7168
30	11610,00	5108,40	7424
		76692,00	

Рис. 17 - Подсчет размера резервных копий

Спустя один месяц резервного копирования на резервном узле будет в общей сложности 85 ГБ резервных данных.

## Этап 2

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

Предположим, что мы потеряли доступ к основному узлу (о нет, я забыла пароль и удалила сообщение от Владимира Вячеславовича с паролем с почты. Какой ужас!)

Удалим все данные из папки кластера на резервном узле.

```
[postgres6@pg169 ~/dyu42]$ rm -rf *
[postgres6@pg169 ~/dyu42]$ ls
[postgres6@pg169 ~/dyu42]$ pg_ctl status -D
pg_ctl: option requires an argument -- D
Для дополнительной информации попробуйте "pg_ctl --help".
[postgres6@pg169 ~/dyu42]$ pg_ctl status -D .
pg_ctl: каталог "." не содержит структуры кластера баз данных
[postgres6@pg169 ~/dyu42]$
```

Рис. 18 - Удаление содержимого dyu42 на резервном узле

Разархивируем базу. Основная часть (не считая созданные дополнительно нами табличные пространства) находится в архиве base.tar.gz. Я сократила скриншот до первых ~10 строк, по-настоящему их вывело больше сотни.

```
[postgres6@pg169 ~/dyu42]$ cd ..
[postgres6@pg169 ~]$ ls backups/base
backup_2024-05-28-17
[postgres6@pg169 ~]$ tar xvzf backups/base/backup_2024-05-28-17/base.tar.gz -C dyu42
x backup_label
x tablespace_map
x pg_serial/
x pg_replslot/
x pg_commit_ts/
x pg_notify/
x pg_dynshmem/
x pg_tblspc/
x tempinsert2
```

Рис. 19 - Восстановление базы данных

После восстановления базы нам необходимо восстановить все WAL-файлы. WAL-файлы входят в состав основной копии, созданной с помощью pg\_basebackup. WAL-файлы находятся в архиве pg\_wal.tar.gz. Разархивируем аналогично самой базе.

```
[postgres6@pg169 ~]$ tar xvzf backups/base/backup_2024-05-28-17/pg_wal.tar.gz -C
x 00000001000000000000000015
```

Рис. 20 - Восстановление WAL-файлов

Это не все. Созданные в лабораторной работе №2 табличные пространства не были восстановлены. Если мы просмотрим содержимое резервной копии, то увидим, что кроме архивов с WAL-файлами и базой есть еще два пронумерованных архива.

```
[postgres6@pg169 ~/backups/backup_2024-05-28-17]$ ls
16384.tar.gz    16385.tar.gz    backup_manifest base.tar.gz      pg_wal.tar.gz
[postgres6@pg169 ~/backups/backup_2024-05-28-17]$
```

Рис. 21 - Содержимое резервной копии

Чтобы разобраться, какой архив принадлежит какому пространству, посмотрим содержимое файла `tablespace_map`, лежащего в директории `dyu42`.

```
insert script pg_ident.conf
[postgres6@pg169 ~/dyu42]$ cat tablespace_map
16384 /var/db/postgres0/sui26
16385 /var/db/postgres0/ipk91
```

Рис. 22 - Нумерация табличных пространств

Создадим директории для табличных пространств аналогично тому, как мы делали это в лабораторной работе №2.

```
[postgres6@pg169 ~]$ mkdir sui26 ipk91
[postgres6@pg169 ~]$ ls
backup_cleanup.sh  backups  dyu42  ipk91  sui26
```

Рис. 23 - Создание директорий для табличных пространств

Разархивируем оба табличных пространства в их директории так же, как мы делали это с основной базой и WAL-файлами.

```
[postgres6@pg169 ~]$ tar xvfz backups/base/backup_2024-05-28-17/16384.tar.gz -C sui26
x PG_14_202107181/
x PG_14_202107181/16386/
[postgres6@pg169 ~]$ tar xvfz backups/base/backup_2024-05-28-17/16385.tar.gz -C ipk91
x PG_14_202107181/
x PG_14_202107181/16386/
[postgres6@pg169 ~]$
```

Рис. 24 - Восстановление табличных пространств

Проверим, что мы сделали все правильно. Для этого сначала запустим сервер.

```
[postgres6@pg169 ~]$ pg_ctl -D $HOME/dyu42 start
ожидание запуска сервера....2024-05-30 13:35:03.559 MSK [49921] LOG:  redirecting log output to logging co
lector process
2024-05-30 13:35:03.559 MSK [49921] HINT:  Future log output will appear in directory "log".
готово
сервер запущен
```

Рис. 25 - Запуск сервера

Теперь попробуем подключиться к базе данных nicepinklake и прочитать данные из какой-нибудь созданной нами таблицы. Как мы видим, все прошло успешно.

```
[postgres6@pg169 ~]$ psql -h localhost -p 9812 -U postgres0 nicepinklake
Пароль пользователя postgres0:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=# select * from lakes
nicepinklake=# ;
 id |   name   | color_id
----+-----+-----
  4 | pinklake |       1
  5 | greenlake|       2
  6 | bluelake |       3
  7 | pinklake |       1
  8 | greenlake|       2
  9 | bluelake |       3
(6 строк)

nicepinklake=#
```

Рис. 26 - Проверка восстановления данных

Восстановление кластера при потере доступа к основному узлу прошло успешно.

## Этап 3

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 конфигурационных файлов СУБД 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.

Предположим, что я магическим образом вспомнила пароль и восстановила доступ к основному узлу, но мой кот прошел по клавиатуре и выполнил следующую команду:

```
[postgres0@pg189 ~]$ rm -r dyu42
[postgres0@pg189 ~]$ ls
070424          basebackups          logfile          postgres0.dump
backup_cleanup.sh  cleanup.sh          pg_backup.sh     sui26
backups          ipk91               pg_wal
```

Рис. 27 - Удаление директории со всем содержимым

Проверим работу СУБД:

```
[postgres0@pg189 ~]$ pg_ctl -D $HOME/dyu42 start
pg_ctl: каталог "/var/db/postgres0/dyu42" не существует
[postgres0@pg189 ~]$ psql -p 9812 -U postgres0 postgres
psql: ошибка: подключиться к серверу через сокет "/tmp/.s.PGSQL.9812" не удалось: No such file or directory
Сервер действительно работает локально и принимает подключения через этот сокет?
[postgres0@pg189 ~]$
```

Рис. 28 - Проверка работы СУБД

Сервер не работает (невозможно запустить), данные недоступны. Проведем восстановление аналогично второму этапу, но теперь из резервной копии, хранящейся на основном узле, но учтем следующее условие: исходное расположение дополнительных табличных пространств недоступно - разместить в другой директории и скорректировать конфигурацию. Предположим, что мы не можем воспользоваться ни директорией sui26, ни директорией ipk91. Создадим новую директорию, в которой будут находиться эти табличные пространства.

```
[postgres0@pg189 ~]$ mkdir -p tablespaces/sui26 tablespaces/ipk91
[postgres0@pg189 ~]$ ls tablespaces
ipk91    sui26
```

Рис. 29 - Создание новой директории для табличных пространств

Восстановим

директорию

dyu42.

```
[postgres0@pg189 ~]$ ls basebackups/backup_2024-05-28-17
16384.tar.gz    16385.tar.gz    backup_manifest base.tar.gz      pg_wal.tar.gz
[postgres0@pg189 ~]$ mkdir dyu42
[postgres0@pg189 ~]$ tar xvzf basebackups/backup_2024-05-28-17/base.tar.gz -C dyu42
x backup_label
x tablespace_map
x pg_serial/
x pg_replslot/
x pg_commit_ts/
x pg_notify/
x pg_dynshmem/
x pg_tblspc/
x tempinsert2
```

Рис. 30 - Восстановление кластера БД

Восстановим табличное пространство sui26.

```
[postgres0@pg189 ~]$ tar xvzf basebackups/backup_2024-05-28-17/16384tar.gz -C tablespaces/sui26
x PG_14_202107181/
x PG_14_202107181/16386/
[postgres0@pg189 ~]$ █
```

Рис. 31 - Восстановление табличного пространства sui26

Восстановим

табличное

пространство

ipk91.

```
[postgres0@pg189 ~]$ tar xvzf basebackups/backup_2024-05-28-17/16385.tar.gz -C tablespaces/ipk91
x PG_14_202107181/
x PG_14_202107181/16386/
```

Рис. 32 - Восстановление табличного пространства ipk91

В восстановленной директории dyu42 указано не актуальное расположение табличных пространств, а недоступное. Изменим расположение с помощью файла tablespace\_map:

```
[postgres0@pg189 ~]$ vi dyu42/tablespace_map
16384 /var/db/postgres0/tablespaces/sui26
16385 /var/db/postgres0/tablespaces/ipk91
~
```

Рис. 33 - Изменение местоположения табличных пространств

Восстановим директорию pg\_wal.

```
[postgres0@pg189 ~]$ tar xvfz basebackups/backup_2024-05-28-17/pg_wal.tar.gz -C dyu42
x 00000001000000000000000015
[postgres0@pg189 ~]$
```

Рис. 34 - Восстановление pg\_wal

Проверим, что в табличных пространствах и директории кластера все восстановилось.

```
[postgres0@pg189 ~]$ ls tablespaces/sui26
PG_14_202107181
[postgres0@pg189 ~]$ ls tablespaces/ipk91
PG_14_202107181
[postgres0@pg189 ~]$ ls dyu42
00000001000000000000000015      pg_dynshmem      pg_tblspc
backup_label                    pg_hba.conf      pg_twophase
base                            pg_ident.conf    PG_VERSION
createscript                    pg_logical        pg_wal
current_logfiles                pg_multixact      pg_xact
global                          pg_notify         postgresql.auto.conf
hba.conf                       pg_replslot      postgresql.conf
insertscript                    pg_serial         tablespace_map
lofgile                         pg_snapshots     tempinsert1
log                             pg_stat           tempinsert2
logfile                         pg_stat_tmp      temptable1
pg_commit_ts                   pg_subtrans      temptable2
```

Рис. 35 - Проверка восстановления директорий

Проверим, что WAL-файлы успешно восстановились.

```
[postgres0@pg189 ~]$ ls dyu42/pg_wal
00000001000000000000000015      archive_status
[postgres0@pg189 ~]$
```

Рис. 36 - Проверка восстановления pg\_wal



Запустим

сервер.

```
chmod: -R: no such file or directory
[postgres0@pg189 ~]$ pg_ctl -D $HOME/duy42 start
ожидание запуска сервера....2024-05-30 14:57:51.512 MSK [59955] LOG:  redirecting log output to logging collector process
2024-05-30 14:57:51.512 MSK [59955] HINT:  Future log output will appear in directory "log".
    готово
сервер запущен
[postgres0@pg189 ~]$
```

Рис. 37 - Запуск сервера

Попробуем прочитать данные из любой таблицы, чтобы убедиться, что восстановление прошло успешно:

```
[postgres0@pg189 ~]$ psql -h localhost -p 9812 -U postgres0 nicepinklake
Пароль пользователя postgres0:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=# select * from people;
 id |   name   | age | occupation 
-----+-----+-----+-----
  1 | john     |  40 | doctor
  2 | mark     |  35 | florist
  3 | michelle |  36 | librarian
  4 | john     |  40 | doctor
  5 | mark     |  35 | florist
  6 | michelle |  36 | librarian
  7 | john     |  40 | doctor
  8 | mark     |  35 | florist
  9 | michelle |  36 | librarian
(9 строк)

nicepinklake=#
```

Рис. 38 - Проверка восстановления данных

Проверим, что директории с табличными пространствами получилось изменить верно. Создадим временную таблицу в пространстве sui26 и проверим, что находится в каждом табличном пространстве:

[illegible]

Рис. 39 - Проверка изменения местоположения табличных пространств

Как мы видим, восстановление на основном узле при потере файлов прошло успешно.

## Этап 4

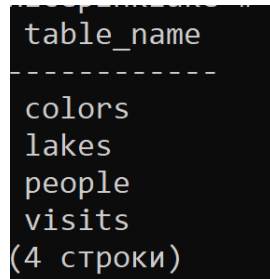
Этот сценарий подразумевает частичную потерю данных (в результате нежелательной или ошибочной операции) при сохранении доступности основного узла. Необходимо выполнить восстановление данных на ОСНОВНОМ узле следующим способом:

- 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:
  - Удалить каждую вторую строку в любой таблице (`DELETE`)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

Проверим, сколько таблиц есть в схеме `public`.



```
table_name
-----
colors
lakes
people
visits
(4 строки)
```

Рис. 40 - Таблицы в схеме `public`

Вставим по три строки в каждую из указанных на рис. 40 таблиц.

```

nicepinklake=# select * from colors;
 id | color
----+-----
  1 | pink
  2 | green
  3 | blue
  4 | pink
  5 | green
  6 | blue
(6 строк)

nicepinklake=# insert into colors (id, color) values
nicepinklake-# jshdlhgdsjkggh
nicepinklake-# ;
ERROR:  syntax error at or near "jshdlhgdsjkggh"
CTPOKA 2: jshdlhgdsjkggh
          ^

nicepinklake=# insert into colors (color) values
nicepinklake-# ('red'), ('white'), ('brown');
INSERT 0 3

```

Рис. 41 - Добавление данных в таблицу colors

```

nicepinklake=# select * from lakes;
 id | name      | color_id
----+-----+-----
  4 | pinklake  |      1
  5 | greenlake |      2
  6 | bluelake  |      3
  7 | pinklake  |      1
  8 | greenlake |      2
  9 | bluelake  |      3
(6 строк)

nicepinklake=# insert into lakes (name, color_id) values
nicepinklake-# ('redlake', 7), ('white', 8), ('brown', 9);
INSERT 0 3

```

Рис. 42 - Добавление данных в таблицу lakes

```
nicepinklake=# select * from people
nicepinklake=# ;
 id |  name  | age | occupation
-----+-----+-----+-----
  1 | john   |  40 | doctor
  2 | mark   |  35 | florist
  3 | michelle | 36 | librarian
  4 | john   |  40 | doctor
  5 | mark   |  35 | florist
  6 | michelle | 36 | librarian
  7 | john   |  40 | doctor
  8 | mark   |  35 | florist
  9 | michelle | 36 | librarian
(9 строк)
```

```
nicepinklake=# insert into people (name, age, occupation) values
nicepinklake-# ('patrick', 25, 'farmer'),
nicepinklake-# ('joseph', 40, 'writer'),
nicepinklake-# ('sarah', 35, 'artist');
INSERT 0 3
```

Рис. 43 - Добавление данных в таблицу people

```
nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 10 |         1 |        5
 11 |         1 |        6
 12 |         2 |        7
 13 |         2 |        5
 14 |         3 |        6
 15 |         3 |        7
(6 строк)
```

```
nicepinklake=# insert into visits(id_human, id_lake) values
nicepinklake-# (8, 6), (9, 7), (10, 8);
INSERT 0 3
```

Рис. 44 - Добавление данных в таблицу visits

Зафиксируем результат. Как и раньше, воспользуемся скриптом с командой pg\_basebackup.

```
[postgres0@pg189 ~]$ bash pg_backup.sh
Password for postgres6@pg169.cs.ifmo.ru:
backup_manifest
16385.tar.gz
base.tar.gz
16384.tar.gz
pg_wal.tar.gz
100% 184KB 104.9MB/s 00:00
100% 142 963.6KB/s 00:00
100% 4513KB 147.9MB/s 00:00
100% 144 1.0MB/s 00:00
100% 17KB 38.9MB/s 00:00
```

Рис. 45 - Создание резервной копии

Зафиксируем в отчете время. Я смотрела время командой date.

```
[postgres0@pg189 ~]$ date
четверг, 30 мая 2024 г. 18:46:55 (MSK)
[postgres0@pg189 ~]$
```

Рис. 46 - Время

По заданию необходимо удалить каждую вторую строку из любой таблицы. Я взяла таблицу visits и удалила из нее все строки с нечетными идентификаторами.

Было:

```
nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 10 |         1 |        5
 11 |         1 |        6
 12 |         2 |        7
 13 |         2 |        5
 14 |         3 |        6
 15 |         3 |        7
 16 |         8 |        6
 17 |         9 |        7
 18 |        10 |        8
(9 строк)
```

Рис. 47 - Таблица visits перед удалением данных

Стало:

```
nicepinklake=# delete from visits where id in (11, 13, 15, 17);
DELETE 4
nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 10 |         1 |        5
 12 |         2 |        7
 14 |         3 |        6
 16 |         8 |        6
 18 |        10 |        8
(5 строк)
```

Рис. 48 - Таблица visits после удаления данных

Восстановим данные на резервном узле так же, как в этапах 2, 3.

```
[postgres@pg169 ~]$ ls backups/base
backup_2024-05-28-17    backup_2024-05-30-20    backup_2024-05-30-45
[postgres@pg169 ~]$ tar xvzf backups/base/backup_2024-05-30-45/base.tar.gz -C dyu42
x backup_label
x tablespace_map
x postgresql.auto.conf
x pg_dynshmem/
x tempinsert2
```

Рис. 49 - Восстановление кластера на резервном узле

```
[postgres@pg169 ~]$ tar xvzf backups/base/backup_2024-05-30-45/pg_wal.tar.gz -C dyu42/pg_wal
x 000000010000000000000001B
[postgres@pg169 ~]$ tar xvzf backups/base/backup_2024-05-30-45/16384.tar.gz -C sui26
tar: could not chdir to 'sui26'

[postgres@pg169 ~]$ tar xvzf backups/base/backup_2024-05-30-45/16384.tar.gz -C tablespaces/sui26
x PG_14_202107181/
x PG_14_202107181/16386/
[postgres@pg169 ~]$ tar xvzf backups/base/backup_2024-05-30-45/16385.tar.gz -C tablespaces/ipk91
x PG_14_202107181/
x PG_14_202107181/16386/
[postgres@pg169 ~]$
```

Рис. 50 - Восстановление табличных пространств, WAL-файлов на резервном узле

Чтобы не получать ошибки, которую я получила раз десять, установим права на папку dyu42 командой `chmod 700 dyu42`. Запустим сервер и проверим, что база восстановилась вместе с новыми добавленными строками.

```
[postgres@pg169 ~]$ chmod 700 dyu42
[postgres@pg169 ~]$ pg_ctl -D $HOME/dyu42 start
ожидание запуска сервера....2024-05-30 19:00:04.797 MSK [88167] LOG:  redirecting log output to logging collector process
2024-05-30 19:00:04.797 MSK [88167] HINT:  Future log output will appear in directory "log".
готово
сервер запущен
[postgres@pg169 ~]$ psql -h localhost -p 9812 -U nicepinkuser nicepinklake
Пароль пользователя nicepinkuser:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=> select * from visits;
 id | id_human | id_lake
-----+-----+-----
 10 |         1 |       5
 11 |         1 |       6
 12 |         2 |       7
 13 |         2 |       5
 14 |         3 |       6
 15 |         3 |       7
 16 |         8 |       6
 17 |         9 |       7
 18 |        10 |       8
(9 строк)

nicepinklake=>
```

Рис. 51 - Запуск сервера на резервном узле

Сделаем dump командой pg\_dump. Команда pg\_dump создает копию одной базы данных. Изменения произошли в базе данных nicepinklake, поэтому в команде укажем именно ее. Для создания копии необходимо, чтобы сервер был включен.

```
[postgres6@pg169 ~]$ pg_dump -c --if-exists -p 9812 -h localhost -U nicepinkuser nicepinklake > dump.sql
Пароль:
[postgres6@pg169 ~]$ ls
backup_cleanup.sh      backups              dump.sql             dyu42                tablespaces
[postgres6@pg169 ~]$
```

Рис. 52 - Создание копии командой pg\_dump

Перенесем созданную на резервном узле копию на основной узел, остановим сервер на резервном узле, так как он нам больше не потребуется, и выйдем из резервного узла.

```
[postgres6@pg169 ~]$ scp dump.sql postgres0@pg189:/var/db/postgres0
Password for postgres0@pg189.cs.ifmo.ru:
dump.sql                                     100% 8058      21.3MB/s   00:00
[postgres6@pg169 ~]$ pg_ctl -D $HOME/dyu42 stop
ожидание завершения работы сервера..... готово
сервер остановлен
[postgres6@pg169 ~]$ exit
выход
Есть остановленные задания.
[postgres6@pg169 ~]$ exit
выход
Connection to pg169.cs.ifmo.ru closed.
[postgres0@pg189 ~]$
```

Рис. 53 - Перенос копии, выход из узла

Восстановим пострадавшую от рук мошенников-уничтожителей базу nicepinklake с помощью копии, отправленной с резервного узла.

```
user postgres0
[postgres0@pg189 ~]$ psql -f dump.sql -p 9812 -h localhost -U nicepinkuser nicepinklake < dump.sql
Пароль пользователя nicepinkuser:
SET
SET
SET
SET
SET
```

Рис. 54 - Восстановление базы данных nicepinklake

Проверим,           получилось           ли           восстановить           утерянные           строки:



```

[postgres0@pg189 ~]$ psql -U nicepinkuser -h localhost -p 9812 nicepinklake
Пароль пользователя nicepinkuser:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=> select * from visits;
 id | id_human | id_lake
-----+-----+-----
 10 |         1 |        5
 11 |         1 |        6
 12 |         2 |        7
 13 |         2 |        5
 14 |         3 |        6
 15 |         3 |        7
 16 |         8 |        6
 17 |         9 |        7
 18 |        10 |        8
(9 строк)

nicepinklake=>

```

Рис. 55 - Проверка восстановления строк

Восстановление базы данных при помощи инструмента `pg_dump` прошло успешно. Остановим сервер и выйдем из основного узла:

```

[postgres0@pg189 ~]$ pg_ctl -D $HOME/dyu42 stop
ожидание завершения работы сервера..... готово
сервер остановлен
[postgres0@pg189 ~]$ exit
выход
Есть остановленные задания.
[postgres0@pg189 ~]$ exit
выход
Connection to pg189 closed.
C:\Users\admin>

```

Рис. 56 - Остановка сервера, выход из узла

## Изменения

На защите я поняла, что сделала ряд лишних действий в четвертом этапе. Переделаю задачу правильно.

Зафиксируем состояние базы данных перед удалением строк командой `pg_dump`.

```
[postgres0@pg189 ~]$ pg_dump -c --if-exists -p 9812 -h localhost -U postgres0 nicepinklake > dump.sql
Пароль:
[postgres0@pg189 ~]$ psql -h localhost -p 9812 -U postgres0 nicepinklake
Пароль пользователя postgres0:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 11 |         1 |        6
 13 |         2 |        5
 15 |         3 |        7
 17 |         9 |        7
 19 |         5 |        5
 21 |         7 |        7
 22 |         5 |        5
 23 |         6 |        6
 24 |         7 |        7
(9 строк)
```

Рис. 57 - Создание резервной копии nicepinklake

Удалим каждую вторую строку:

```
nicepinklake=# delete from visits where id in (11, 15, 19, 22, 24);
DELETE 5
nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 13 |         2 |        5
 17 |         9 |        7
 21 |         7 |        7
 23 |         6 |        6
(4 строки)

nicepinklake=# ^Z
[4]+  Остановлен      psql -h localhost -p 9812 -U postgres0 nicepinklake
```

Рис. 58 - Повреждение данных

Восстановим базу:

```
[postgres0@pg189 ~]$ psql -f dump.sql -p 9812 -h localhost -U postgres0 nicepinklake < dump.sql
Пароль пользователя postgres0:
SET
SET
SET
SET
SET
set config
```

Рис. 59 - Восстановление из dump.sql

Проверим, что база восстановилась успешно:

```
[postgres0@pg189 ~]$ psql -h localhost -p 9812 -U postgres0 nicepinklake
Пароль пользователя postgres0:
psql (14.2)
Введите "help", чтобы получить справку.

nicepinklake=# select * from visits;
 id | id_human | id_lake
-----+-----+-----
 11 |         1 |        6
 13 |         2 |        5
 15 |         3 |        7
 17 |         9 |        7
 19 |         5 |        5
 21 |         7 |        7
 22 |         5 |        5
 23 |         6 |        6
 24 |         7 |        7
(9 строк)

nicepinklake=#
```

Рис. 60 - Проверка успешного восстановления

## **Выводы**

В ходе лабораторной работы я научилась создавать логические и физические резервные копии баз данных при помощи команд `pg_basebackup`, `pg_dump`. Я смогла смитировать повреждение файловой системы, потерю доступа к базе данных и случайное удаление строк из баз данных и сумела восстановить кластер и базы данных во всех перечисленных случаях.