Университет ИТМО

Факультет программной инженерии и компьютерной техники

Распределённые системы хранения данных. Лабораторная работа №3.

Группа: P33131
Студент: Смирнов Виктор Игоревич
Преподаватель: Афанасьев Дмитрий Борисович
Вариант: 736

2024

Ключевые слова

База данных, конфигурация PostgreSQL.

Содержание

1 Цель работы и контекст **1**

2 Этап 0. Контекст работы **1**
2.1 Переменные окружения . . . 2
2.2 Конфигурация базы данных . . . 2
2.3 Создание .pgpass . . . 3
2.4 Инициализация базы данных . . . 3
2.5 Запуск базы данных . . . 4
2.6 Настройка базы данных . . . 4

3 Этап 1. Резервное копирование **4**
3.1 Задача . . . 4
3.2 Подготовка секретов . . . 5
3.3 Конфигурация primary узла для резервного копирования . . . 5
3.4 Создание базовой резервной копии . . . 5
3.5 Подготовка standby . . . 6
3.6 Полная настройка primary узла . . . 6
3.7 Действия системного администратора по первоначальной настройке системы . . . 6

4 Этап 2. Потеря основного узла **7**
4.1 Задача . . . 7
4.2 Восстановление СУБД на резервном узле . . . 7
4.3 Действия на primary узле . . . 8
4.4 Действия на standby узле . . . 8

5 Этап 3. Повреждение файлов БД **9**
5.1 Задача . . . 9
5.2 Решение . . . 10

6 Этап 4. Логическое повреждение данных **11**
6.1 Задача . . . 11
6.2 Фиксируем состояние БД . . . 12
6.3 Восстановление дампа из бэкапа на резервном узле . . . 12

7 Вывод **12**

1 Цель работы и контекст

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.

2 Этап 0. Контекст работы

В предыдущей лабораторной работы была создана база данных. Приведу здесь скрипты для ее инициализации.

2.1 Переменные окружения

```
#!/bin/sh

export DDB_PG_CONF="."
export DDB_PG_USER="postgres0"
export DDB_PG_PASS="pleasehelp"
export DDB_PG_PASS_FILE="$DDB_PG_CONF/pgpass.txt"
export DDB_PG_PORT=9666
export DDB_PG_DATABASE=postgres

export DDB_TABLESPACE_NAME=yqy90
export DDB_TABLESPACE_LOCATION="$HOME/$DDB_TABLESPACE_NAME"
export DDB_NEW_DATABASE_NAME=lazyorangehair
export DDB_NEW_USER=root
export DDB_NEW_USER_PASSWORD=rootik

export PGDATA="$HOME/kop67"

export DDB_PG_BIN_DIR=/usr/lib/postgresql/14/bin
export DDB_INITDB=$DDB_PG_BIN_DIR/initdb
export DDB_PGBIN=$DDB_PG_BIN_DIR/postgres
export DDB_PGBASEBACKUP=$DDB_PG_BIN_DIR/pg_basebackup
export DDB_PGCTL=$DDB_PG_BIN_DIR/pg_ctl
export DDB_PGVERIFYBACKUP=$DDB_PG_BIN_DIR/pg_verifybackup
export DDB_PGDUMP=$DDB_PG_BIN_DIR/pg_dump
export DDB_PGRESTORE=$DDB_PG_BIN_DIR/pg_restore

export DDB_BACKUP_DIR="primary/backup"
export DDB_BACKUP_BASE_DIR="$DDB_BACKUP_DIR/base"
export DDB_BACKUP_WAL_DIR="$DDB_BACKUP_DIR/wal"
export DDB_BACKUP_DUMP_DIR="$DDB_BACKUP_DIR/dump"

export DDB_STANDBY_HOST=ddb-standby
export DDB_STANDBY_USER=$DDB_PG_USER
```

Листинг 1: Переменные окружения

2.2 Конфигурация базы данных

```
# $PGDATA/pg_hba.conf (Host-based authentication)

# TYPE  DATABASE    USER    ADDRESS        METHOD
host    all         all     127.0.0.1/32   scram-sha-256 # Permit only localhost
host    all         all     ::1/128        scram-sha-256 # Permit only localhost
host    replication all     localhost      scram-sha-256 # Permit base backup
```

Листинг 2: Конфигурационный файл pg_hba.conf

```
# $PGDATA/postgresql.conf (PostgreSQL configuration file)

# Note: Optimized for OLAP load:
# 5 users, packet r/w 128MB

## CONNECTIONS

listen_addresses  = '127.0.0.1' # Available only from localhost
port              = 9666        # For security
unix_socket_directories = ''    # Only TCP/IP

max_connections                 = 6 # 5 users + 1 extra
superuser_reserved_connections  = 3


## AUTHENTICATION

authentication_timeout  = 20s           # Type password faster
password_encryption     = scram-sha-256 # Strong password hashing

## RESOURCE USAGE

shared_buffers            = 1024MB  # 128MB * (5 + 3) users
temp_buffers              = 128MB   # 128MB
```

```
max_prepared_transactions = 0        # We don't use transactions
work_mem                  = 256MB    # Expected packet size
hash_mem_multiplier       = 1.5      # Smaller hash tables
maintenance_work_mem      = 64MB     # ?
autovacuum_work_mem       = -1
max_stack_depth           = 4MB      # Be prepared for complex queries

temp_file_limit           = 4GB      # Something is wrong if we reach this

## WRITE-AHEAD LOG

checkpoint_timeout    = 5min
fsync                 = off          # Lost data is not critical, as we can recreate
synchronous_commit    = off          # Same
wal_level             = replica      # Enable replication
wal_compression       = off          # WAL must not be so huge?
commit_delay          = 200          # Acceptable to lose 200mc of data
effective_cache_size  = 4GB          # OK?


## REPORTING AND LOGGING

log_destination    = 'stderr'
logging_collector = off
log_directory      = 'log'
log_filename       = 'postgresql-%Y-%m-%d_%H%M%S.log'
log_min_messages   = warning

log_connections    = on
log_disconnections = on

## Archiving

archive_mode    = on
archive_timeout = 16s
archive_command = 'ssh -q <STANDBY_HOST> "test ! -e <STANDBY_WAL_DIR>/%f" && scp %p <
    STANDBY_HOST>:~/<STANDBY_WAL_DIR>'
```

Листинг 3: Конфигурационный файл postgresql.conf

2.3 Создание .pgpass

```
#!/bin/sh

set -e

cd "$(dirname "$0")"

echo "" > ~/.pgpass
echo "localhost:$DDB_PG_PORT:*:$DDB_PG_USER:$DDB_PG_PASS" >> ~/.pgpass
echo "localhost:$DDB_PG_PORT:*:$DDB_NEW_USER:$DDB_NEW_USER_PASSWORD" >> ~/.pgpass
chmod 0600 ~/.pgpass
```

Листинг 4: Файл .pgpass

2.4 Инициализация базы данных

```
#!/bin/sh

set -e

cd "$(dirname "$0")"

mkdir "$PGDATA" 2> /dev/null

echo "$DDB_PG_PASS" > "$DDB_PG_PASS_FILE"

"$DDB_INITDB" \
  --pgdata="$PGDATA" \
  --locale="ru_RU.CP1251" \
  --encoding="WIN1251" \
  --pwfile="$DDB_PG_PASS_FILE"

```

```
cp "$DDB_PG_CONF/pg_hba.conf" "$PGDATA/pg_hba.conf"
cp "$DDB_PG_CONF/postgresql.conf" "$PGDATA/postgresql.conf"
```

Листинг 5: Инициализация базы данных

2.5 Запуск базы данных

```
#!/bin/sh

set -e

cd "$(dirname "$0")"

"$DDB_PGBIN" -D "$PGDATA"
```

Листинг 6: Запуск базы данных

2.6 Настройка базы данных

```
#!/bin/sh

set -e

cd "$(dirname "$0")"

sql() {
    psql -h localhost -p "$DDB_PG_PORT" -c "$1" "$DDB_PG_DATABASE"
}

mkdir "$DDB_TABLESPACE_LOCATION" 2>/dev/null

sql "CREATE TABLESPACE $DDB_TABLESPACE_NAME LOCATION '$DDB_TABLESPACE_LOCATION';"
sql "ALTER DATABASE template1 SET TABLESPACE $DDB_TABLESPACE_NAME;"
sql "CREATE DATABASE $DDB_NEW_DATABASE_NAME TEMPLATE template1;"
sql "CREATE ROLE tester;"
sql "CREATE USER $DDB_NEW_USER WITH LOGIN PASSWORD '$DDB_NEW_USER_PASSWORD';"
sql "GRANT tester TO $DDB_NEW_USER;"

sql() {
  psql -U "$DDB_NEW_USER" -h localhost -p $DDB_PG_PORT -c "$2" "$1"
}

PRV="$DDB_PG_DATABASE"
NEW="$DDB_NEW_DATABASE_NAME"

sql "$PRV" "CREATE TABLE note_prv (id serial PRIMARY KEY, content text NOT NULL);"
sql "$NEW" "CREATE TABLE note_new (id serial PRIMARY KEY, content text NOT NULL);"

sql "$PRV" "INSERT INTO note_prv (content) VALUES ('Note at postgres');"
sql "$NEW" "INSERT INTO note_new (content) VALUES ('Note at lazyorangehair');"

sql "$PRV" "SELECT * FROM note_prv;"
sql "$NEW" "SELECT * FROM note_new;"
```

Листинг 7: Настройка базы данных

3 Этап 1. Резервное копирование

3.1 Задача

1. Настроить резервное копирование с основного узла на резервный следующим образом:
   (a) Первоначальная полная копия + непрерывное архивирование.
   (b) Включить для СУБД режим архивирования WAL;
   (c) настроить копирование WAL (scp) на резервный узел;
   (d) создать первоначальную резервную копию (pg_ basebackup),
   (e) скопировать на резервный узел (rsync).
2. Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:

(a) Средний объем новых данных в БД за сутки: 650МБ.

(b) Средний объем измененных данных за сутки: 950МБ.

3. Проанализировать результаты.

3.2 Подготовка секретов

Нам необходимо будет отправлять базовую резервную копию, а так WAL файлы на резервный узел, так что сперва следует сгенерировать и распределить ключи шифрования для безопасной передачи данных между узлами.

```
#!/bin/sh

set -e

echo "[primary] Generating ssh key..."
ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ""

echo "[primary] Generated ssh key:"
cat ~/.ssh/id_rsa.pub
```

Листинг 8: Генерация ключей

Далее я авторизовал публичный ключ primary узла на standby. Теперь можно проверить, что передача данных скорее всего будет работать.

```
#!/bin/sh

set -e

ssh -q $1 "echo Hello, World!"
```

Листинг 9: Проверка подключения

3.3 Конфигурация primary узла для резервного копирования

Включаем архивирование, будем отправлять WAL файлы на standby каждые 16 секунд.

```
## Archiving

archive_mode    = on
archive_timeout = 16s
archive_command = 'ssh -q <STANDBY_HOST> "test ! -e <STANDBY_WAL_DIR>/%f" && scp %p <
    STANDBY_HOST>:~/<STANDBY_WAL_DIR>'
```

Листинг 10: Ключевые строчки в конфигурационном файле

3.4 Создание базовой резервной копии

```
#!/bin/sh

set -e

echo "[primary] Creating base backup..."
"$DDB_PGBASEBACKUP" \
    --host="localhost" \
    --port="$DDB_PG_PORT" \
    --pgdata="$HOME/$DDB_BACKUP_BASE_DIR" \
    --format="tar" \
    --wal-method="fetch" \
    --no-password

echo "[primary] Sending to '$DDB_STANDBY_HOST'..."
rsync -ave ssh \
    "$HOME/$DDB_BACKUP_BASE_DIR" \
    $DDB_STANDBY_USER@$DDB_STANDBY_HOST:~/$DDB_BACKUP_DIR
```

Листинг 11: Создание базовой резервной копии

3.5 Подготовка standby

```
#!/bin/sh

set -e

echo "[standby] Prepairing..."

mkdir -p "$HOME/$DDB_BACKUP_WAL_DIR"
mkdir -p "$HOME/$DDB_BACKUP_BASE_DIR"
mkdir -p "$HOME/$DDB_BACKUP_DUMP_DIR"
```

Листинг 12: Подготовка standby

3.6 Полная настройка primary узла

```
#!/bin/sh

set -e

echo "[primary] Creating '.pgpass' file..."
sh common/pgpass.sh

echo "[primary] Editing 'postgresql.conf' file..."
sh primary/config.sh

echo "[primary] Initializing the database..."
sh primary/init.sh

echo "[primary] Starting the database..."
sh common/start.sh &

echo "[primary] Waiting the database startup..."
sleep 2

echo "[primary] Settings up the database..."
sh primary/setup.sh

echo "[primary] All right!"
```

Листинг 13: Полная настройка primary узла

3.7 Действия системного администратора по первоначальной настройке системы

1. Получить конфигурационные файлы системы из репозитория `https://github.com/vityaman-edu/ddb-homework/tree/trunk/lab-3`
2. Доставить директории `db/common` и `db/primary` на `primary` узел, разместив их в домашней директории пользователя, от лица которого будет запущена система
3. Доставить директории `db/common` и `db/standby` на `standby` узел, разместив их в домашней директории пользователя, от лица которого будет запущена система
4. На `primary` узле сгенерировать ключи для `primary` узла при помощи скрипта `common/ssh-keygen.sh` и авторизовать публичный ключ на узле standby
5. Проверить на `primary` узле возможность ssh соединения с `standby` узлом при помощи `common/ssh-test.sh`
6. Подготовить `standby` узел к резервированию, выполнив на нем `source common/env.sh && sh standby/prepare.sh`
7. Запустить `primary` узел, выполнив на нем `source common/env.sh && sh common/pgpass.sh && sh primary/full.sh`
8. Создать базовую резервную копию на `primary` узле и отправить ее на `standby` узел: `sh primary/backup.sh`
9. Наполнить данными базу данных на `primary` узле: `sh primary/fill.sh`
10. Убедиться, что базовая резервная копия и WAL файлы доставлены на `standby` узел

4 Этап 2. Потеря основного узла

4.1 Задача

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

4.2 Восстановление СУБД на резервном узле

Для этого необходимо просто выполнить `source common/env.sh && sh standby/restore.sh` на `standby` узле, предварительно убедившись, что в директории `primary/backup/base` находятся файлы базовой резервной копии, а в директории `primary/backup/wal` есть WAL сегменты.

```
#!/bin/sh

# set -e

MODE=$1
echo "[restore] MODE: $MODE"

TARGET_TIME=$2
echo "[restore] TARGET_TIME: $TARGET_TIME"

echo "[restore] Removing existing database installation..."
sh common/clear.sh

mkdir -p $PGDATA
chmod 0700 $PGDATA

echo "[restore] Extracting base backup..."
tar \
    --extract \
    -f "$HOME/$DDB_BACKUP_BASE_DIR/base.tar" \
    --directory=$PGDATA

echo "[restore] Restoring tablespaces..."
while read -r line; do
    TABLESPACE_OID=$(echo $line | awk '{print $1}')

    if [ "$MODE" = "anon-tblspc" ]; then
        TABLESPACE_DIR="$HOME/tablespace/$TABLESPACE_OID"
    else
        TABLESPACE_DIR=$(echo $line | awk '{print $2}')
    fi

    echo "[restore][$TABLESPACE_OID] Restoring tablespace..."

    echo "[restore][$TABLESPACE_OID] Directory: $TABLESPACE_DIR"
    mkdir -p $TABLESPACE_DIR

    echo "[restore][$TABLESPACE_OID] Extracting..."
    tar --extract \
        -f "$HOME/$DDB_BACKUP_BASE_DIR/$TABLESPACE_OID.tar" \
        --directory=$TABLESPACE_DIR

    echo "[restore][$TABLESPACE_OID] Creating symbolic link..."
    ln -s $TABLESPACE_DIR $PGDATA/pg_tblspc/$TABLESPACE_OID
done <$PGDATA/tablespace_map

echo "[restore] Checking base backup integrity..."
"$DDB_PGVERIFYBACKUP" \
    --manifest-path="$HOME/$DDB_BACKUP_BASE_DIR/backup_manifest" \
    --wal-directory="$HOME/$DDB_BACKUP_WAL_DIR" \
    $PGDATA

echo "[restore] Removing 'tablespace_map'..."
rm $PGDATA/tablespace_map

# Patching is used as we need to change values depending on env variables
echo "[restore] Patching postgresql.conf: add 'restore_command'..."
```

```
RESTORE_CMD="restore_command = 'cp ~/$DDB_BACKUP_WAL_DIR/%f %p'"
echo "\n$RESTORE_CMD\n" >> $PGDATA/postgresql.conf

if [ "$MODE" = "standby" ]; then
    echo "[restore] Patching postgresql.conf: disable archive..."
    sed -i -e "s+archive_mode+#archive_mode+g" $PGDATA/postgresql.conf
fi

if [ "$TARGET_TIME" != "" ]; then
    echo "[restore] Patching postgresql.conf: setting target time..."
    echo "\nrecovery_target_time = '$TARGET_TIME'\n" >> $PGDATA/postgresql.conf
    echo "\nrecovery_target_inclusive = false\n"     >> $PGDATA/postgresql.conf
fi

echo "[restore] Signalling of recovery..."
touch $PGDATA/recovery.signal

echo "[restore] Is ready for startup!"
```

Листинг 14: Восстановление СУБД

4.3 Действия на primary узле

```
postgres0@2d09031f584d:~$ history
1  ls
2  source common/env.sh
3  sh common/ssh-keygen.sh
4  sh common/ssh-test.sh ddb-standby
5  sh common/pgpass.sh
6  sh primary/full.sh
7  sh primary/backup.sh
8  sh primary/fill.sh
9  history
```

Листинг 15: Действия на primary узле

4.4 Действия на standby узле

```
postgres0@c33884c20d42:~$ history
2  ls
3  source common/env.sh
4  vim .ssh/authorized_keys
5  sh standby/prepare.sh
6  ls
7  ls primary/backup/base/
8  ls primary/backup/wal/
9  ls primary/backup/base
10  ls primary/backup/wal/
11  sh standby/restore.sh
12  sh common/start.sh
13  bg
14  history
```

Листинг 16: Действия на standby узле

```
postgres0@c33884c20d42:~$ sh common/start.sh
2024-05-28 08:56:18.375 GMT [1088] LOG:  starting PostgreSQL 14.12 (Ubuntu 14.12-1.
    pgdg20.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2)
     9.4.0, 64-bit
2024-05-28 08:56:18.375 GMT [1088] LOG:  listening on IPv4 address "127.0.0.1", port
    9666
2024-05-28 08:56:18.378 GMT [1089] LOG:  database system was interrupted; last known up
    at 2024-05-28 08:50:55 GMT
cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
2024-05-28 08:56:18.381 GMT [1089] LOG:  starting archive recovery
2024-05-28 08:56:18.392 GMT [1089] LOG:  restored log file "000000010000000000000002"
    from archive
2024-05-28 08:56:18.396 GMT [1089] LOG:  redo starts at 0/2000028
2024-05-28 08:56:18.396 GMT [1089] LOG:  consistent recovery state reached at 0/2000138
```

```
2024-05-28 08:56:18.397 GMT [1088] LOG:  database system is ready to accept read-only
    connections
2024-05-28 08:56:18.406 GMT [1089] LOG:  restored log file "000000010000000000000003"
    from archive
2024-05-28 08:56:18.417 GMT [1089] LOG:  restored log file "000000010000000000000004"
    from archive
cp: cannot stat '/home/postgres0/primary/backup/wal/000000010000000000000005': No such
    file or directory
2024-05-28 08:56:18.420 GMT [1089] LOG:  redo done at 0/4000148 system usage: CPU: user:
     0.00 s, system: 0.00 s, elapsed: 0.02 s
2024-05-28 08:56:18.420 GMT [1089] LOG:  last completed transaction was at log time
    2024-05-28 08:51:07.961265+00
2024-05-28 08:56:18.430 GMT [1089] LOG:  restored log file "000000010000000000000004"
    from archive
cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
2024-05-28 08:56:18.435 GMT [1089] LOG:  selected new timeline ID: 2
2024-05-28 08:56:18.441 GMT [1089] LOG:  archive recovery complete
cp: cannot stat '/home/postgres0/primary/backup/wal/00000001.history': No such file or
    directory
2024-05-28 08:56:18.450 GMT [1088] LOG:  database system is ready to accept connections
```

Листинг 17: Вывод postgres при старте

```
$ psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -d postges
postgres=# select * from note_prv;
 id |          content
----+--------------------------
  1 | Note at postgres
  2 | Another note at postgres
(2 rows)

$ psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -d lazyorangehair
lazyorangehair=# select * from note_new;
 id |             content
----+---------------------------------
  1 | Note at lazyorangehair
  2 | Another note at lazyorangehair
(2 rows)
```

Листинг 18: Состояние СУБД на standby узле

Как мы видим, в базе данных сохранились не только данные с базовой копии, но и подтянулись изменения из WAL сегментов.

5 Этап 3. Повреждение файлов БД

5.1 Задача

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:

1. Симулировать сбой: удалить с диска директорию любой таблицы со всем содержимым.
2. Проверить работу СУБД, доступность данных, перезапустить СУБД, проанализировать результаты.
3. Выполнить восстановление данных из резервной копии, учитывая следующее условие: исходное расположение дополнительных табличных пространств недоступно - разместить в другой директории и скорректировать конфигурацию.
4. Запустить СУБД, проверить работу и доступность данных, проанализировать результаты.

5.2 Решение

Попробуем удалить таблицу `note_prv`. Для этого узнаем, где представлены ее данные в файловой системе.

```
$ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
postgres=# SELECT pg_relation_filepath('note_prv');
 pg_relation_filepath
----------------------
 base/14374/16389
(1 row)
```

Листинг 19: Получаем физическую локацию таблицы

Начинаем уничтожение.

```
$ rm $PGDATA/base/14374/16389
```

Листинг 20: Удаляем файл таблицы

Наблюдаем эффекты.

```
$ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
postgres=# select * from note_prv;
ERROR:  could not open file "base/14374/16389": No such file or directory
```

Листинг 21: Проверяем работоспособность СУБД

Попробуем перезапустить базу данных.

```
$ sh common/stop.sh
waiting for server to shut down.... done
server stopped

$ sh common/start.sh
2024-05-28 10:52:21.905 GMT [9200] LOG:  starting PostgreSQL 14.12 (Ubuntu 14.12-1.
    pgdg20.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2)
     9.4.0, 64-bit
2024-05-28 10:52:21.905 GMT [9200] LOG:  listening on IPv4 address "127.0.0.1", port
    9666
2024-05-28 10:52:21.908 GMT [9206] LOG:  database system was shut down at 2024-05-28
    10:52:16 GMT
2024-05-28 10:52:21.916 GMT [9200] LOG:  database system is ready to accept connections

$ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
postgres=# select * from note_prv;
2024-05-28 10:54:18.117 GMT [9230] ERROR:  could not open file "base/14374/16389": No
    such file or directory
2024-05-28 10:54:18.117 GMT [9230] STATEMENT:  select * from note_prv;
ERROR:  could not open file "base/14374/16389": No such file or directory
```

Листинг 22: Перезапускаем датабазу

Видим, что postgres не заметил пропажи, значит не проверил целостность файлов. Наверное, должен существовать способ осуществить проверку целостность его файлов. `pg_checksums` не обнаружил проблему. По идее, можно использовать для проверки утилиту `pg_verifybackup`, которая проверит, соответствует ли содержимое `PGDATA` заданному бэкапу.

Скачиваем бэкап с `standby`, сносим нашу больную датабазу, запускаем восстановление и проверяем результат.

```
$ sh primary/download.sh

$ ls primary/backup/base primary/backup/wal/
primary/backup/base:
16384.tar  backup_manifest  base.tar

primary/backup/wal/:
000000010000000000000001
000000010000000000000002
000000010000000000000002.00000028.backup
000000010000000000000003
000000010000000000000004
000000010000000000000005
$ sh common/clear.sh

```

```
$ sh common/restore.sh

$ sh common/start.sh
2024-05-28 11:36:31.487 GMT [13049] LOG:  starting PostgreSQL 14.12 (Ubuntu 14.12-1.pgdg20.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0, 64-bit
2024-05-28 11:36:31.487 GMT [13049] LOG:  listening on IPv4 address "127.0.0.1", port 9666
2024-05-28 11:36:31.489 GMT [13051] LOG:  database system was interrupted; last known up at 2024-05-28 08:50:55 GMT
cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or directory
2024-05-28 11:36:31.492 GMT [13051] LOG:  starting archive recovery
2024-05-28 11:36:31.502 GMT [13051] LOG:  restored log file "000000010000000000000002" from archive
2024-05-28 11:36:31.506 GMT [13051] LOG:  redo starts at 0/2000028
2024-05-28 11:36:31.506 GMT [13051] LOG:  consistent recovery state reached at 0/2000138
2024-05-28 11:36:31.507 GMT [13049] LOG:  database system is ready to accept read-only connections
2024-05-28 11:36:31.516 GMT [13051] LOG:  restored log file "000000010000000000000003" from archive
2024-05-28 11:36:31.527 GMT [13051] LOG:  restored log file "000000010000000000000004" from archive
2024-05-28 11:36:31.539 GMT [13051] LOG:  restored log file "000000010000000000000005" from archive
cp: cannot stat '/home/postgres0/primary/backup/wal/000000010000000000000006': No such file or directory
2024-05-28 11:36:31.541 GMT [13051] LOG:  redo done at 0/50001C0 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.03 s
2024-05-28 11:36:31.541 GMT [13051] LOG:  last completed transaction was at log time 2024-05-28 08:51:07.961265+00
2024-05-28 11:36:31.552 GMT [13051] LOG:  restored log file "000000010000000000000005" from archive
cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or directory
2024-05-28 11:36:31.557 GMT [13051] LOG:  selected new timeline ID: 2
2024-05-28 11:36:31.563 GMT [13051] LOG:  archive recovery complete
cp: cannot stat '/home/postgres0/primary/backup/wal/00000001.history': No such file or directory
2024-05-28 11:36:31.573 GMT [13049] LOG:  database system is ready to accept connections

$ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
postgres=# select * from note_prv;
 id |          content
----+--------------------------
  1 | Note at postgres
  2 | Another note at postgres
(2 rows)
```

Листинг 23: Восстанавливаем базу данных

Починили.

Вторая часть выполняется аналогично, но нужно вызывать восстановление через `sh common/restore.bash anon-tblspc`.

6 Этап 4. Логическое повреждение данных

6.1 Задача

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:

1. В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
2. Зафиксировать время и симулировать ошибку: в любой таблице с внешними ключами подменить значения ключей на случайные (INSERT, UPDATE)

3. Продемонстрировать результат.
4. Выполнить восстановление данных указанным способом.
5. Продемонстрировать и проанализировать результат.

6.2 Фиксируем состояние БД

Сперва запустим вставку нескольких строк на основном узле через скрипт, после чего фиксируем текущее время.

```
#!/bin/sh

set -e

cd "$(dirname "$0")"

sql() {
    psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -c "$1" $DDB_PG_DATABASE
}

sql "INSERT INTO note_prv (content) VALUES ('The first testing note');"
sql "INSERT INTO note_prv (content) VALUES ('The second testing note');"
sql "INSERT INTO note_prv (content) VALUES ('The third testing note');"
```

Листинг 24: Вставка дополнительных строк

6.3 Восстановление дампа из бэкапа на резервном узле

7 Вывод

Данная лабораторная работа помогла мне изучить конфигурацию PostgreSQL.

Список литературы

[1] PostgreSQL Documentation: сайт. - 2024. - URL: https://www.postgresql.org/docs/14/index.html (дата обращения: 06.04.2024) - Текст : электронный.