

Университет ИТМО
Факультет программной инженерии и компьютерной техники

Распределённые системы хранения данных. Лабораторная работа №3.

Группа: Р33131
Студент: Смирнов Виктор Игоревич
Преподаватель: Афанасьев Дмитрий Борисович
Вариант: 736

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 <code>pgbase</code>	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 <code>primary</code> узла для резервного копирования	5
3.4	Создание базовой резервной копии	5
3.5	Подготовка <code>standby</code>	5
3.6	Полная настройка <code>primary</code> 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 <code>primary</code> узле	8
4.4	Действия на <code>standby</code> 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	Решение	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 работе была создана база данных. Приведу здесь скрипты для ее инициализации.

2.1 Переменные окружения

```
1 #!/bin/sh
2
3 export DDB_PG_CONF="."
4 export DDB_PG_USER="postgres0"
5 export DDB_PG_PASS="pleasehelp"
6 export DDB_PG_PASS_FILE="$DDB_PG_CONF/pgpass.txt"
7 export DDB_PG_PORT=9666
8 export DDB_PG_DATABASE=postgres
9
10 export DDB_TABLESPACE_NAME=yqy90
11 export DDB_TABLESPACE_LOCATION="$HOME/$DDB_TABLESPACE_NAME"
12 export DDB_NEW_DATABASE_NAME=lazyorangehair
13 export DDB_NEW_USER=root
14 export DDB_NEW_USER_PASSWORD=rootik
15
16 export PGDATA="$HOME/kop67"
17
18 export DDB_PG_BIN_DIR=/usr/lib/postgresql/14/bin
19 export DDB_INITDB=$DDB_PG_BIN_DIR/initdb
20 export DDB_PG_BIN=$DDB_PG_BIN_DIR/postgres
21 export DDB_PG_BASEBACKUP=$DDB_PG_BIN_DIR/pg_basebackup
22 export DDB_PG_CTL=$DDB_PG_BIN_DIR/pg_ctl
23 export DDB_PG_VERIFYBACKUP=$DDB_PG_BIN_DIR/pg_verifybackup
24 export DDB_PGDUMP=$DDB_PG_BIN_DIR/pg_dump
25 export DDB_PGRESTORE=$DDB_PG_BIN_DIR/pg_restore
26
27 export DDB_BACKUP_DIR="primary/backup"
28 export DDB_BACKUP_BASE_DIR="$DDB_BACKUP_DIR/base"
29 export DDB_BACKUP_WAL_DIR="$DDB_BACKUP_DIR/wal"
30 export DDB_BACKUP_DUMP_DIR="$DDB_BACKUP_DIR/dump"
31
32 export DDB_STANDBY_HOST=ddb-standby
```

Листинг 1: Переменные окружения

2.2 Конфигурация базы данных

```
1 # $PGDATA/pg_hba.conf (Host-based authentication)
2
3 # TYPE      DATABASE      USER      ADDRESS      METHOD
4 host       all           all       127.0.0.1/32  scram-sha-256 # Permit only localhost
5 host       all           all       ::1/128      scram-sha-256 # Permit only localhost
6 host       replication all       localhost    scram-sha-256 # Permit base backup
```

Листинг 2: Конфигурационный файл pg_hba.conf

```
1 # $PGDATA/postgresql.conf (PostgreSQL configuration file)
2
3 # Note: Optimized for OLAP load:
4 # 5 users, packet r/w 128MB
5
6 ## CONNECTIONS
7
8 listen_addresses = '127.0.0.1' # Available only from localhost
9 port             = 9666        # For security
10 unix_socket_directories = ''   # Only TCP/IP
11
12 max_connections   = 6 # 5 users + 1 extra
13 superuser_reserved_connections = 3
14
15
16 ## AUTHENTICATION
17
18 authentication_timeout = 20s # Type password faster
19 password_encryption    = scram-sha-256 # Strong password hashing
20
21 ## RESOURCE USAGE
22
23 shared_buffers        = 1024MB # 128MB * (5 + 3) users
24 temp_buffers          = 128MB   # 128MB
25 max_prepared_transactions = 0    # We don't use transactions
```

```

26 work_mem                = 256MB      # Expected packet size
27 hash_mem_multiplier     = 1.5        # Smaller hash tables
28 maintenance_work_mem   = 64MB       # ?
29 autovacuum_work_mem     = -1
30 max_stack_depth         = 4MB        # Be prepared for complex queries
31
32 temp_file_limit         = 4GB        # Something is wrong if we reach this
33
34 ## WRITE-AHEAD LOG
35
36 checkpoint_timeout      = 5min
37 fsync                   = off        # Lost data is not critical, as we can recreate
38 synchronous_commit      = off        # Same
39 wal_level               = replica    # Enable replication
40 wal_compression         = off        # WAL must not be so huge?
41 commit_delay            = 200        # Acceptable to lose 200mc of data
42 effective_cache_size    = 4GB       # OK?
43
44
45 ## REPORTING AND LOGGING
46
47 log_destination         = 'stderr'
48 logging_collector       = off
49 log_directory           = 'log'
50 log_filename            = 'postgresql-%Y-%m-%d_%H%M%S.log'
51 log_min_messages        = warning
52
53 log_connections         = on
54 log_disconnections      = on
55
56 ## Archiving
57
58 archive_mode            = on
59 archive_timeout         = 16s
60 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

```

1 #!/bin/sh
2
3 set -e
4
5 cd "$(dirname "$0")"
6
7 echo "" > ~/.pgpass
8 echo "localhost:$DDB_PG_PORT:*$DDB_PG_USER:$DDB_PG_PASS" >> ~/.pgpass
9 echo "localhost:$DDB_PG_PORT:*$DDB_NEW_USER:$DDB_NEW_USER_PASSWORD" >> ~/.pgpass
10 chmod 0600 ~/.pgpass

```

Листинг 4: Файл .pgpass

2.4 Инициализация базы данных

```

1 #!/bin/sh
2
3 set -e
4
5 cd "$(dirname "$0")"
6
7 mkdir "$PGDATA" 2> /dev/null
8
9 echo "$DDB_PG_PASS" > "$DDB_PG_PASS_FILE"
10
11 "$DDB_INITDB" \
12 --pgdata="$PGDATA" \
13 --locale="ru_RU.CP1251" \
14 --encoding="WIN1251" \
15 --pwfile="$DDB_PG_PASS_FILE"
16
17 cp "$DDB_PG_CONF/pg_hba.conf" "$PGDATA/pg_hba.conf"

```

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

Листинг 5: Инициализация базы данных

2.5 Запуск базы данных

```
1 #!/bin/sh
2
3 set -e
4
5 cd "$(dirname "$0")"
6
7 "$DDB_PGBIN" -D "$PGDATA"
```

Листинг 6: Запуск базы данных

2.6 Настройка базы данных

```
1 #!/bin/sh
2
3 set -e
4
5 cd "$(dirname "$0")"
6
7 sql() {
8     psql -h localhost -p "$DDB_PG_PORT" -c "$1" "$DDB_PG_DATABASE"
9 }
10
11 mkdir "$DDB_TABLESPACE_LOCATION" 2>/dev/null
12
13 sql "CREATE TABLESPACE $DDB_TABLESPACE_NAME LOCATION '$DDB_TABLESPACE_LOCATION';"
14 sql "ALTER DATABASE template1 SET TABLESPACE $DDB_TABLESPACE_NAME;"
15 sql "CREATE DATABASE $DDB_NEW_DATABASE_NAME TEMPLATE template1;"
16 sql "CREATE ROLE tester;"
17 sql "CREATE USER $DDB_NEW_USER WITH LOGIN PASSWORD '$DDB_NEW_USER_PASSWORD';"
18 sql "GRANT tester TO $DDB_NEW_USER;"
19
20 sql() {
21     psql -U "$DDB_NEW_USER" -h localhost -p $DDB_PG_PORT -c "$2" "$1"
22 }
23
24 PRV="$DDB_PG_DATABASE"
25 NEW="$DDB_NEW_DATABASE_NAME"
26
27 sql "$PRV" "CREATE TABLE note_prv (id serial PRIMARY KEY, content text NOT NULL);"
28 sql "$NEW" "CREATE TABLE note_new (id serial PRIMARY KEY, content text NOT NULL);"
29
30 sql "$PRV" "INSERT INTO note_prv (content) VALUES ('Note at postgres');"
31 sql "$NEW" "INSERT INTO note_new (content) VALUES ('Note at lazyorangehair');"
32
33 sql "$PRV" "SELECT * FROM note_prv;"
34 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.

```
1 #!/bin/sh
2
3 set -e
4
5 echo "[primary] Generating ssh key..."
6 ssh-keygen -t rsa -f ~/.ssh/id_rsa -N ""
7
8 echo "[primary] Generated ssh key:"
9 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.

```
1 #!/bin/sh
2
3 set -e
4
5 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.

```
1 ## Archiving
2
3 archive_mode = on
4 archive_timeout = 16s
5 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

```
1 #!/bin/sh
2
3 set -e
4
5 echo "[primary] Creating base backup..."
6 "$DDB_PGBASEBACKUP" \
7   --host="localhost" \
8   --port="$DDB_PG_PORT" \
9   --pgdata="$HOME/$DDB_BACKUP_BASE_DIR" \
10  --format="tar" \
11  --wal-method="fetch" \
12  --no-password
13
14 echo "[primary] Sending to '$DDB_STANDBY_HOST'..."
15 rsync -ave ssh "$HOME/$DDB_BACKUP_BASE_DIR" $DDB_STANDBY_HOST:~/DDB_BACKUP_DIR
```

Листинг 11: Создание базовой резервной копии

3.5 Подготовка standby

```

1 #!/bin/sh
2
3 set -e
4
5 echo "[standby] Preparing..."
6
7 mkdir -p "$HOME/$DDB_BACKUP_WAL_DIR"
8 mkdir -p "$HOME/$DDB_BACKUP_BASE_DIR"
9 mkdir -p "$HOME/$DDB_BACKUP_DUMP_DIR/$DDB_PG_DATABASE"
10 mkdir -p "$HOME/$DDB_BACKUP_DUMP_DIR/$DDB_NEW_DATABASE_NAME"

```

Листинг 12: Подготовка standby

3.6 Полная настройка primary узла

```

1 #!/bin/sh
2
3 set -e
4
5 echo "[primary] Creating '.pgpass' file..."
6 sh common/0-pgpass.sh
7
8 echo "[primary] Editing 'postgresql.conf' file..."
9 sh primary/1-config.sh
10
11 echo "[primary] Initializing the database..."
12 sh primary/1-init.sh
13
14 echo "[primary] Starting the database..."
15 sh common/2-start.sh &
16
17 echo "[primary] Waiting the database startup..."
18 sleep 2
19
20 echo "[primary] Settings up the database..."
21 sh primary/3-setup.sh
22
23 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/0-ssh-keygen.sh и авторизовать публичный ключ на узле standby
5. Проверить на primary узле возможность ssh соединения с standby узлом при помощи common/0-ssh-test.sh
6. Подготовить standby узел к резервированию, выполнив на нем source common/0-env.sh && sh standby/1-prepare.sh
7. Запустить primary узел, выполнив на нем source common/0-env.sh && sh common/0-pgpass.sh && sh primary/9-full.sh
8. Создать базовую резервную копию на primary узле и отправить ее на standby узел: sh primary/4-backup.sh
9. Наполнить данными базу данных на primary узле: sh primary/5-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/0-env.sh && sh standby/2-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.

```
1 #!/bin/sh
2
3 # set -e
4
5 MODE=$1
6 echo "[restore] MODE: $MODE"
7
8 TARGET_TIME=$2
9 echo "[restore] TARGET_TIME: $TARGET_TIME"
10
11 echo "[restore] Removing existing database installation..."
12 sh common/3-clear.sh
13
14 mkdir -p $PGDATA
15 chmod 0700 $PGDATA
16
17 echo "[restore] Extracting base backup..."
18 tar \
19   --extract \
20   -f "$HOME/$DDB_BACKUP_BASE_DIR/base.tar" \
21   --directory=$PGDATA
22
23 echo "[restore] Restoring tablespaces..."
24 while read -r line; do
25     TABLESPACE_OID=$(echo $line | awk '{print $1}')
26
27     if [ "$MODE" = "anon-tblspc" ]; then
28         TABLESPACE_DIR="$HOME/tablespace/$TABLESPACE_OID"
29     else
30         TABLESPACE_DIR=$(echo $line | awk '{print $2}')
31     fi
32
33     echo "[restore][$TABLESPACE_OID] Restoring tablespace..."
34
35     echo "[restore][$TABLESPACE_OID] Directory: $TABLESPACE_DIR"
36     mkdir -p $TABLESPACE_DIR
37
38     echo "[restore][$TABLESPACE_OID] Extracting..."
39     tar --extract \
40       -f "$HOME/$DDB_BACKUP_BASE_DIR/$TABLESPACE_OID.tar" \
41       --directory=$TABLESPACE_DIR
42
43     echo "[restore][$TABLESPACE_OID] Creating symbolic link..."
44     ln -s $TABLESPACE_DIR $PGDATA/pg_tblspc/$TABLESPACE_OID
45 done <$PGDATA/tablespace_map
46
47 echo "[restore] Checking base backup integrity..."
48 "$DDB_PGVERIFYBACKUP" \
49   --manifest-path="$HOME/$DDB_BACKUP_BASE_DIR/backup_manifest" \
50   --wal-directory="$HOME/$DDB_BACKUP_WAL_DIR" \
51   $PGDATA
52
53 echo "[restore] Removing 'tablespace_map'..."
54 rm $PGDATA/tablespace_map
55
56 # Patching is used as we need to change values depending on env variables
57 echo "[restore] Patching postgresql.conf: add 'restore_command'..."
```



```

58 RESTORE_CMD="restore_command = 'cp ~/$DDB_BACKUP_WAL_DIR/%f %p'"
59 echo "\n$RESTORE_CMD\n" >> $PGDATA/postgresql.conf
60
61 if [ "$MODE" = "standby" ]; then
62     echo "[restore] Patching postgresql.conf: disable archive..."
63     sed -i -e "s+archive_mode+#archive_mode+g" $PGDATA/postgresql.conf
64 fi
65
66 if [ "$TARGET_TIME" != "" ]; then
67     echo "[restore] Patching postgresql.conf: setting target time..."
68     echo "\nrecovery_target_time = '$TARGET_TIME'\n" >> $PGDATA/postgresql.conf
69     echo "\nrecovery_target_inclusive = false\n" >> $PGDATA/postgresql.conf
70 fi
71
72 echo "[restore] Signalling of recovery..."
73 touch $PGDATA/recovery.signal
74
75 echo "[restore] Is ready for startup!"

```

Листинг 14: Восстановление СУБД

4.3 Действия на primary узле

```

1 postgres0@2d09031f584d:~$ history
2 1 ls
3 2 source common/0-env.sh
4 3 sh common/0-ssh-keygen.sh
5 4 sh common/0-ssh-test.sh ddb-standby
6 5 sh common/0-pgpass.sh
7 6 sh primary/9-full.sh
8 7 sh primary/4-backup.sh
9 8 sh primary/5-fill.sh
10 9 history

```

Листинг 15: Действия на primary узле

4.4 Действия на standby узле

```

1 postgres0@c33884c20d42:~$ history
2 2 ls
3 3 source common/0-env.sh
4 4 vim .ssh/authorized_keys
5 5 sh standby/1-prepare.sh
6 6 ls
7 7 ls primary/backup/base/
8 8 ls primary/backup/wal/
9 9 ls primary/backup/base
10 10 ls primary/backup/wal/
11 11 sh standby/2-restore.sh
12 12 sh common/2-start.sh
13 13 bg
14 14 history

```

Листинг 16: Действия на standby узле

```

1 postgres0@c33884c20d42:~$ sh common/2-start.sh
2 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
3 2024-05-28 08:56:18.375 GMT [1088] LOG:  listening on IPv4 address "127.0.0.1", port
    9666
4 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
5 cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
6 2024-05-28 08:56:18.381 GMT [1089] LOG:  starting archive recovery
7 2024-05-28 08:56:18.392 GMT [1089] LOG:  restored log file "000000010000000000000002"
    from archive
8 2024-05-28 08:56:18.396 GMT [1089] LOG:  redo starts at 0/2000028
9 2024-05-28 08:56:18.396 GMT [1089] LOG:  consistent recovery state reached at 0/2000138

```

```

10 2024-05-28 08:56:18.397 GMT [1088] LOG:  database system is ready to accept read-only
    connections
11 2024-05-28 08:56:18.406 GMT [1089] LOG:  restored log file "000000010000000000000003"
    from archive
12 2024-05-28 08:56:18.417 GMT [1089] LOG:  restored log file "000000010000000000000004"
    from archive
13 cp: cannot stat '/home/postgres0/primary/backup/wal/000000010000000000000005': No such
    file or directory
14 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
15 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
16 2024-05-28 08:56:18.430 GMT [1089] LOG:  restored log file "000000010000000000000004"
    from archive
17 cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
18 2024-05-28 08:56:18.435 GMT [1089] LOG:  selected new timeline ID: 2
19 2024-05-28 08:56:18.441 GMT [1089] LOG:  archive recovery complete
20 cp: cannot stat '/home/postgres0/primary/backup/wal/00000001.history': No such file or
    directory
21 2024-05-28 08:56:18.450 GMT [1088] LOG:  database system is ready to accept connections

```

Листинг 17: Вывод postgres при старте

```

1 $ psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -d postgres
2 postgres=# select * from note_prv;
3  id |          content
4  ----+-----
5    1 | Note at postgres
6    2 | Another note at postgres
7 (2 rows)
8
9 $ psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -d lazyorangehair
10 lazyorangehair=# select * from note_new;
11  id |          content
12  ----+-----
13    1 | Note at lazyorangehair
14    2 | Another note at lazyorangehair
15 (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.

```
1 $ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
2 postgres=# SELECT pg_relation_filepath('note_prv');
3 pg_relation_filepath
4 -----
5 base/14374/16389
6 (1 row)
```

Листинг 19: Получаем физическую локацию таблицы

Начинаем уничтожение.

```
1 $ rm $PGDATA/base/14374/16389
```

Листинг 20: Удаляем файл таблицы

Наблюдаем эффекты.

```
1 $ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
2 postgres=# select * from note_prv;
3 ERROR:  could not open file "base/14374/16389": No such file or directory
```

Листинг 21: Проверяем работоспособность СУБД

Попробуем перезапустить базу данных.

```
1 $ sh common/4-stop.sh
2 waiting for server to shut down.... done
3 server stopped
4
5 $ sh common/2-start.sh
6 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
7 2024-05-28 10:52:21.905 GMT [9200] LOG:  listening on IPv4 address "127.0.0.1", port
   9666
8 2024-05-28 10:52:21.908 GMT [9206] LOG:  database system was shut down at 2024-05-28
   10:52:16 GMT
9 2024-05-28 10:52:21.916 GMT [9200] LOG:  database system is ready to accept connections
10
11 $ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
12 postgres=# select * from note_prv;
13 2024-05-28 10:54:18.117 GMT [9230] ERROR:  could not open file "base/14374/16389": No
   such file or directory
14 2024-05-28 10:54:18.117 GMT [9230] STATEMENT:  select * from note_prv;
15 ERROR:  could not open file "base/14374/16389": No such file or directory
```

Листинг 22: Перезапускаем базу

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 большую базу, запускаем восстановление и проверяем результат.

```
1 $ sh primary/6-download.sh
2
3 $ ls primary/backup/base primary/backup/wal/
4 primary/backup/base:
5 16384.tar backup_manifest base.tar
6
7 primary/backup/wal/:
8 00000001000000000000000000000001
9 00000001000000000000000000000002
10 00000001000000000000000000000002.00000028.backup
11 00000001000000000000000000000003
12 00000001000000000000000000000004
13 00000001000000000000000000000005
14 $ sh common/3-clear.sh
15
```

```

16 $ sh common/1-restore.sh
17
18 $ sh common/2-start.sh
19 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
20 2024-05-28 11:36:31.487 GMT [13049] LOG:  listening on IPv4 address "127.0.0.1", port
    9666
21 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
22 cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
23 2024-05-28 11:36:31.492 GMT [13051] LOG:  starting archive recovery
24 2024-05-28 11:36:31.502 GMT [13051] LOG:  restored log file "000000010000000000000002"
    from archive
25 2024-05-28 11:36:31.506 GMT [13051] LOG:  redo starts at 0/2000028
26 2024-05-28 11:36:31.506 GMT [13051] LOG:  consistent recovery state reached at 0/2000138
27 2024-05-28 11:36:31.507 GMT [13049] LOG:  database system is ready to accept read-only
    connections
28 2024-05-28 11:36:31.516 GMT [13051] LOG:  restored log file "000000010000000000000003"
    from archive
29 2024-05-28 11:36:31.527 GMT [13051] LOG:  restored log file "000000010000000000000004"
    from archive
30 2024-05-28 11:36:31.539 GMT [13051] LOG:  restored log file "000000010000000000000005"
    from archive
31 cp: cannot stat '/home/postgres0/primary/backup/wal/000000010000000000000006': No such
    file or directory
32 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
33 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
34 2024-05-28 11:36:31.552 GMT [13051] LOG:  restored log file "000000010000000000000005"
    from archive
35 cp: cannot stat '/home/postgres0/primary/backup/wal/00000002.history': No such file or
    directory
36 2024-05-28 11:36:31.557 GMT [13051] LOG:  selected new timeline ID: 2
37 2024-05-28 11:36:31.563 GMT [13051] LOG:  archive recovery complete
38 cp: cannot stat '/home/postgres0/primary/backup/wal/00000001.history': No such file or
    directory
39 2024-05-28 11:36:31.573 GMT [13049] LOG:  database system is ready to accept connections
40
41 $ psql -h localhost -p "$DDB_PG_PORT" "$DDB_PG_DATABASE"
42 postgres=# select * from note_prv;
43  id |
44  ---+-----
45   1 | Note at postgres
46   2 | Another note at postgres
47 (2 rows)

```

Листинг 23: Восстанавливаем базу данных

Починили.

Вторая часть выполняется аналогично, но нужно вызывать восстановление через `sh common/1-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 Решение

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

```
1 #!/bin/sh
2
3 set -e
4
5 cd "$(dirname "$0")"
6
7 sql() {
8     psql -U "$DDB_PG_USER" -h localhost -p $DDB_PG_PORT -c "$1" $DDB_PG_DATABASE
9 }
10
11 sql "INSERT INTO note_prv (content) VALUES ('The first testing note');"
12 sql "INSERT INTO note_prv (content) VALUES ('The second testing note');"
13 sql "INSERT INTO note_prv (content) VALUES ('The third testing note');"
```

Листинг 24: Вставка дополнительных строк

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.