

Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №3

По дисциплине «Распределённые системы хранения данных» (семестр 6)

Студент:

Дениченко Александр Р3312

Практик:

Осипов Святослав

Санкт-Петербург
2025 г.

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 выдаёт 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.

Изначально сделаем донастройку конфигураций прежней базы данных.

Листинг 1: kitty

```
1 docker create --name postgres-cont-1 -e POSTGRES_PASSWORD=root -p 9193:9193 postgres
```

Листинг 2: kitty

```
1 docker start postgres-cont-1
2 docker exec -it postgres-cont-1 /bin/bash
```

Изменение некоторых настроек разрешений

Листинг 3: postgresql.conf

```
1 listen_addresses = '*'
```

Листинг 4: pg_hba.conf

```
1 host      all      all      0.0.0.0/0      scram-sha-256
```

Подключение к первому узлу теперь выглядит следующим образом.

Листинг 5: kitty

```
1 psql -h 127.0.0.1 -p 9193 -U postgres -d postgres
2 psql -h 127.0.0.1 -p 9193 -U postgres -d fatrednews
3 psql -h 127.0.0.1 -p 9193 -U fatreduser -d fatrednews
```

postgres - root
fatreduser - changeMe

Табличные пространства из прошлой лабы

Листинг 6: kitty

List of tablespaces						
Name	Owner	Location	Access privileges	Options	Size	
Description						
pg_default	postgres				22 MB	
pg_global	postgres				565	
sgk31	postgres	/var/lib/postgresql/sgk31			0	
bytes						

```

7 | yrp30      | postgres | /var/lib/postgresql/yrp30 |      | 0
   | bytes      |          |                             |      |
8 | yva58      | postgres | /var/lib/postgresql/yva58 |      | 0
   | bytes      |          |                             |      |
9 | (5 rows)

```

Создан узел для хранения бэкапов

Листинг 7: kitty

```

1 | docker create --name postgres-backup -p 9194:9193 ubuntu:latest tail -f /dev/null

```

Добавлены утилиты на резервном узле и на основном

Листинг 8: kitty

```

1 | apt-get install -y cron openssh-client openssh-server gzip

```

Была сделана сеть докер для обмена

Листинг 9: kitty

```

1 | docker network create --driver bridge postgres-backup-net
2 | docker network connect postgres-backup-net postgres-cont-1 postgres-backup
3 | docker restart postgres-cont-1 postgres-backup

```

Добавим конфиг в сервер

Листинг 10: kitty

```

1 | echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
2 | chmod 700 /root/.ssh
3 | mkdir -p /run/sshd
4 | chmod 755 /run/sshd
5 | service ssh start

```

Добавлены ключи для упрощения обмена данными на основном сервере

Листинг 11: kitty

```

1 | ssh-keygen -t rsa -b 4096
2 | cat ~/.ssh/id_rsa.pub

```

И добавили ключи в резервный узел

Листинг 12: kitty

```

1 | vim ~/.ssh/authorized_keys ....

```

Настройка скрипта для копирования

Листинг 13: kitty

```

1 | #!/bin/bash
2 |
3 | PG_USER="postgres"
4 | BACKUP_DIR="/tmp/backups"
5 | REMOTE_HOST="help"
6 | REMOTE_DIR="/backups"
7 | TIMESTAMP=$(date +%Y%m%d_%H%M%S)
8 | FULL_BACKUP_FILE="full_backup-$TIMESTAMP.sql.gz"
9 | TABLESPACE_INFO_FILE="tablespace_info-$TIMESTAMP.txt"
10 | LOG_FILE="/tmp/backup_log.txt"
11 |
12 | echo "Starting backup: $TIMESTAMP" >> $LOG_FILE

```

```

13 mkdir -p $BACKUP_DIR
14
15 echo "Creating database dump..." >> $LOG_FILE
16 if pg_dumpall -p 9193 -U $PG_USER | gzip > $BACKUP_DIR/$FULL_BACKUP_FILE; then
17     echo "Backup created successfully: $FULL_BACKUP_FILE" >> $LOG_FILE
18 else
19     echo "Backup failed" >> $LOG_FILE
20     exit 1
21 fi
22
23 echo "Collecting tablespace information..." >> $LOG_FILE
24 psql -p 9193 -U $PG_USER -c "SELECT spcname, pg_catalog.pg_tablespace_location(oid) FROM
    pg_catalog.pg_tablespace;" > $BACKUP_DIR/$TABLESPACE_INFO_FILE
25 echo "Tablespace information collected" >> $LOG_FILE
26
27 echo "Transferring files to backup server..." >> $LOG_FILE
28 if scp $BACKUP_DIR/$FULL_BACKUP_FILE $BACKUP_DIR/$TABLESPACE_INFO_FILE
    $REMOTE_HOST:$REMOTE_DIR/; then
29     echo "Backup transferred successfully" >> $LOG_FILE
30 else
31     echo "Transfer failed" >> $LOG_FILE
32     exit 1
33 fi
34
35 rm $BACKUP_DIR/$FULL_BACKUP_FILE $BACKUP_DIR/$TABLESPACE_INFO_FILE
36
37 echo "Backup script completed" >> $LOG_FILE

```

Сама настройка для автоматизации на основном хосте

Листинг 14: kitty

```

1 crontab -e
2 */1 * * * * /backup.sh
3 cron

```

Настройка для автоматизации на доп хосте

Листинг 15: kitty

```

1 crontab -e
2 */2 * * * * /cleanup_dumps.sh
3 cron

```

Расчет объема резервных копий через месяц

Исходные данные:

- Средний объем новых данных в БД за сутки: 950 МБ.
- Средний объем измененных данных за сутки: 150 МБ.
- Период хранения резервных копий: 4 недели (28 дней).

Предположения:

1. Бэкап делается полностью (включает все данные, а не только новые или измененные).
2. Измененные данные не увеличивают общий объем бэкапа, так как они уже входят в полную копию.
3. Объем БД увеличивается со временем за счет новых данных.

Каждый день создается новая полная резервная копия. Объем бэкапа на n-й день равен всему объему базы данных на тот момент.

Объем базы через n дней можно выразить как:

$$V(n) = V_0 + 950 \times n$$

Где:

- V_0 — начальный объем базы (пусть 0 для расчета за 1 месяц),
- 950 — рост базы в день.

Общий объем всех бэкапов за 28 дней:

$$V_{\text{total}} = \sum_{n=1}^{28} (950 \times n)$$

—

Рассчитаем сумму:

$$V_{\text{total}} = 950 \times (1 + 2 + \dots + 28)$$

Сумма арифметической прогрессии:

$$S = \frac{n(n+1)}{2}$$

Где $n = 28$:

$$S = \frac{28 \times 29}{2} = 406$$

Подставляем:

$$V_{\text{total}} = 950 \times 406 = 385700 \text{ МБ} = 385.7 \text{ ГБ}$$

Потеря основного узла

Листинг 16: kitty

```

1 #!/bin/bash
2
3 set -e
4
5 PGDATA=${PGDATA:-"/var/lib/postgresql/data"}
6 BACKUP_DIR="/backups"
7 PORT=9193
8 TBSP_BASE="/var/lib/postgresql"
9
10 TABLESPACES=("yva58" "yrp30" "sgk31")
11
12 if ! command -v pg_ctl && /dev/null; then
13     PG_PATHS=(
14         "/usr/lib/postgresql/17/bin"
15         "/usr/lib/postgresql/*/bin"
16         "/usr/pgsql-17/bin"
17         "/usr/pgsql-*/bin"
18         "/opt/postgresql/17/bin"
19         "/opt/postgresql/*/bin"
20     )
21
22     for pg_path in "${PG_PATHS[@]}; do
23         for possible_path in $pg_path; do
24             if [ -d "$possible_path" ] && [ -x "$possible_path/pg_ctl" ]; then
25                 export PATH="$possible_path:$PATH"
26                 break 2
27             fi

```

```

28         done
29     done
30
31     if ! command -v pg_ctl &> /dev/null; then
32         exit 1
33     fi
34 fi
35
36 LATEST_BACKUP=$(ls -t ${BACKUP_DIR}/full_backup-*.sql.gz 2>/dev/null | head -1)
37 if [ -z "$LATEST_BACKUP" ]; then
38     exit 1
39 fi
40
41
42 AVAILABLE_LOCALES=$(locale -a)
43 if echo "$AVAILABLE_LOCALES" | grep -q "en_US.utf8"; then
44     REPLACEMENT_LOCALE="en_US.utf8"
45 elif echo "$AVAILABLE_LOCALES" | grep -q "en_US.UTF-8"; then
46     REPLACEMENT_LOCALE="en_US.UTF-8"
47 elif echo "$AVAILABLE_LOCALES" | grep -q "C.UTF-8"; then
48     REPLACEMENT_LOCALE="C.UTF-8"
49 else
50     REPLACEMENT_LOCALE="C"
51 fi
52
53 TEMP_DUMP=$(mktemp)
54
55 gunzip -c "$LATEST_BACKUP" | sed \
56     -e "s/LOCALE = 'en_US.utf8'/LOCALE = '$REPLACEMENT_LOCALE'/g" \
57     -e "s/LOCALE = 'ru_RU.utf8'/LOCALE = '$REPLACEMENT_LOCALE'/g" \
58     -e "s/LOCALE = 'en_US.UTF-8'/LOCALE = '$REPLACEMENT_LOCALE'/g" \
59     -e "s/LOCALE = 'ru_RU.UTF-8'/LOCALE = '$REPLACEMENT_LOCALE'/g" \
60     > "$TEMP_DUMP"
61
62 for tbspace in "${TABLESPACES[@]}"; do
63     tbspace_dir="$TBSP_BASE/$tbspace"
64     mkdir -p "$tbspace_dir"
65     chown -R postgres:postgres "$tbspace_dir"
66     chmod 700 "$tbspace_dir"
67 done
68
69 if pg_ctl -D "$PGDATA" status > /dev/null 2>&1; then
70     pg_ctl -D "$PGDATA" stop -m fast
71 fi
72
73 if [ -d "$PGDATA" ]; then
74     rm -rf "${PGDATA:?}"/*
75 fi
76
77 if ! command -v initdb &> /dev/null; then
78     exit 1
79 fi
80
81 initdb -D "$PGDATA" --locale="$REPLACEMENT_LOCALE" || {
82     exit 1
83 }

```

```

84
85 echo "port = $PORT" >> "$PGDATA/postgresql.conf"
86 echo "listen_addresses = '*' >> "$PGDATA/postgresql.conf"
87 echo "host all all all md5" >> "$PGDATA/pg_hba.conf"
88
89
90 pg_ctl -D "$PGDATA" -o "-p $PORT" start || {
91     if [ -f "$PGDATA/log/postgresql.log" ]; then
92         tail "$PGDATA/log/postgresql.log"
93     elif [ -d "$PGDATA/log" ]; then
94         find "$PGDATA/log" -type f -name "*.log" | xargs tail
95     fi
96     exit 1
97 }
98
99 if ! pg_ctl -D "$PGDATA" status > /dev/null 2>&1; then
100     exit 1
101 fi
102
103 cat "$TEMP_DUMP" | psql -p $PORT -U postgres postgres
104
105 if ! psql -p $PORT -U postgres -lqt | cut -d \\\ -f 1 | grep -qw fatrednews; then
106     psql -p $PORT -U postgres -c "CREATE DATABASE fatrednews WITH TEMPLATE = template0
107         ENCODING = 'UTF8' LOCALE_PROVIDER = libc LOCALE = '$REPLACEMENT_LOCALE';"
108     psql -p $PORT -U postgres -c "ALTER DATABASE fatrednews OWNER TO postgres;"
109 fi
110
111 rm -f "$TEMP_DUMP"
112
113 sleep 5
114
115 pg_ctl -D "$PGDATA" restart || {
116     exit 1
117 }
118
119 psql -p $PORT -U postgres -c "\l"

```

Пример восстановления базы данных

Запуск резервного узла

Листинг 17: kitty

```
1 docker exec -it postgres-backup /bin/bash
```

Должно быть включено ssh и общая есть у двух контейнеров.

Листинг 18: kitty

```
1 service ssh start
```

Делаем типом последний бэкап с основного узла перед его выходом из строя

Листинг 19: kitty

```

1 root@44dfe2ea5829:/# ./backup.sh
2     full_backup-20250403_162858.sql.gz          100% 1699      10.1MB/s    00:00
3     tablespace_info-20250403_162858.txt        100%  242        3.0MB/s    00:00

```

Восстанавливаем кластер на резервном узле

Листинг 20: kitty

```
1 root@49f86d293fa3:/# sudo -u postgres bash /restore.sh
```

Повреждение файлов БД

По факту мы можем применять скрипт для основного узла в любом состоянии бд, так как резервная копия находится на резервном узле, просто происходит запрос в скрипте и развёртывание нового кластера в новой директории

Листинг 21: kitty

```
1 root@44dfe2ea5829:/# ./restore.sh
```

Сам скрипт

Листинг 22: kitty

```
1 #!/bin/bash
2
3
4 PG_USER="postgres"
5 REMOTE_HOST="help"
6 REMOTE_BACKUP_DIR="/backups"
7 LOCAL_TEMP_DIR="/tmp/restore"
8 NEW_PGDATA="/var/lib/postgresql/new_data"
9 PORT=9193
10 LOG_FILE="/tmp/restore_log.txt"
11 POSTGRES_CONF="/etc/postgresql/15/main/postgresql.conf"
12
13 TABLESPACES=("yva58" "yrp30" "sgk31")
14 TBSP_BASE="/var/lib/postgresql"
15
16
17 mkdir -p $LOCAL_TEMP_DIR
18
19 LATEST_BACKUP=$(ssh $REMOTE_HOST "ls -t $REMOTE_BACKUP_DIR/full_backup-*.sql.gz | head -1")
20 LATEST_TABLESPACE_INFO=$(ssh $REMOTE_HOST "ls -t $REMOTE_BACKUP_DIR/tablespace_info-*.txt |
21     head -1")
22
23 if [ -z "$LATEST_BACKUP" ]; then
24     exit 1
25 fi
26
27 BACKUP_FILENAME=$(basename "$LATEST_BACKUP")
28 TABLESPACE_FILENAME=$(basename "$LATEST_TABLESPACE_INFO")
29
30 scp "$REMOTE_HOST:$LATEST_BACKUP" "$LOCAL_TEMP_DIR/$BACKUP_FILENAME"
31 scp "$REMOTE_HOST:$LATEST_TABLESPACE_INFO" "$LOCAL_TEMP_DIR/$TABLESPACE_FILENAME"
32
33 systemctl stop postgresql || {
34     su - postgres -c "pg_ctl stop -D /var/lib/postgresql/15/main -m fast"
35 }
36
37 mkdir -p $NEW_PGDATA
38 chown postgres:postgres $NEW_PGDATA
39 chmod 700 $NEW_PGDATA
40
```



```

41 for tbspace in "${TABLESPACES[@]}"; do
42     tbspace_dir="$TBSP_BASE/$tbspace"
43     mkdir -p "$tbspace_dir"
44     chown postgres:postgres "$tbspace_dir"
45     chmod 700 "$tbspace_dir"
46 done
47
48 AVAILABLE_LOCALES=$(locale -a)
49 if echo "$AVAILABLE_LOCALES" | grep -q "en_US.utf8"; then
50     LOCALE="en_US.utf8"
51 elif echo "$AVAILABLE_LOCALES" | grep -q "en_US.UTF-8"; then
52     LOCALE="en_US.UTF-8"
53 elif echo "$AVAILABLE_LOCALES" | grep -q "C.UTF-8"; then
54     LOCALE="C.UTF-8"
55 else
56     LOCALE="C"
57 fi
58
59 TEMP_DUMP=$(mktemp)
60
61 gunzip -c "$LOCAL_TEMP_DIR/$BACKUP_FILENAME" | sed \
62     -e "s/LOCALE = 'en_US.utf8' /LOCALE = '$LOCALE' /g" \
63     -e "s/LOCALE = 'ru_RU.utf8' /LOCALE = '$LOCALE' /g" \
64     -e "s/LOCALE = 'en_US.UTF-8' /LOCALE = '$LOCALE' /g" \
65     -e "s/LOCALE = 'ru_RU.UTF-8' /LOCALE = '$LOCALE' /g" \
66     > "$TEMP_DUMP"
67
68 su - postgres -c "initdb -D $NEW_PGDATA --locale=$LOCALE" || {
69     exit 1
70 }
71
72
73 sed -i "s|data_directory = '.*'|data_directory = '$NEW_PGDATA'|" $POSTGRES_CONF
74
75 systemctl start postgresql || {
76     su - postgres -c "pg_ctl start -D $NEW_PGDATA -o '-p $PORT'" || {
77         exit 1
78     }
79 }
80
81 su - postgres -c "cat $TEMP_DUMP | psql -p $PORT"
82
83 if ! su - postgres -c "psql -p $PORT -lqt" | cut -d \|| -f 1 | grep -qw fatrednews; then
84     su - postgres -c "psql -p $PORT -c \"CREATE DATABASE fatrednews WITH TEMPLATE =
85         template0 ENCODING = 'UTF8' LOCALE_PROVIDER = libc LOCALE = '$LOCALE';\""
86     su - postgres -c "psql -p $PORT -c \"ALTER DATABASE fatrednews OWNER TO postgres;\""
87 fi
88
89 rm -f "$TEMP_DUMP"
90
91 systemctl restart postgresql || {
92     su - postgres -c "pg_ctl restart -D $NEW_PGDATA -o '-p $PORT'" || {
93         exit 1
94     }
95 }

```

```

96 if systemctl is-active postgresql >/dev/null; then
97 else
98     if su - postgres -c "pg_ctl status -D $NEW_PGDATA" >/dev/null; then
99     else
100         exit 1
101     fi
102 fi
103
104 su - postgres -c "psql -p $PORT -c '\l'" | tee -a $LOG_FILE

```

Логическое повреждение данных

1. Подготовка среды для архивирования WAL

Сначала я создал директорию для хранения WAL архивов:

Листинг 23: kitty

```

1  mkdir -p /var/lib/postgresql/wal_archive
2  chown postgres:postgres /var/lib/postgresql/wal_archive

```

Затем настроил параметры WAL архивирования в PostgreSQL:

Листинг 24: kitty

```

1  su - postgres -c "psql -p 9193 -c \"ALTER SYSTEM SET wal_level = 'replica';\""
2  su - postgres -c "psql -p 9193 -c \"ALTER SYSTEM SET archive_mode = 'on';\""
3  su - postgres -c "psql -p 9193 -c \"ALTER SYSTEM SET archive_command = 'cp %p
   /var/lib/postgresql/wal_archive/%f';\""

```

После этого перезапустил PostgreSQL для применения настроек:

Листинг 25: kitty

```

1  su - postgres -c "pg_ctl restart -D /var/lib/postgresql/data -o '-p 9193'"

```

2. Создание базовой резервной копии

Создал директорию для хранения базовой резервной копии:

Листинг 26: kitty

```

1  mkdir -p /tmp/pg_basebackup
2  chown postgres:postgres /tmp/pg_basebackup

```

Выполнил базовое резервное копирование:

Листинг 27: kitty

```

1  su - postgres -c "pg_basebackup -D /tmp/pg_basebackup -p 9193 -X stream -c fast -P"

```

3. Добавление тестовых данных

Создал тестовые таблицы для демонстрации:

Листинг 28: kitty

```

1  su - postgres -c "psql -p 9193 -c \"
2  CREATE TABLE IF NOT EXISTS test_parent (
3      id SERIAL PRIMARY KEY,
4      name VARCHAR(100),
5      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
6  );
7
8  CREATE TABLE IF NOT EXISTS test_child (

```

```

9      id SERIAL PRIMARY KEY,
10     parent_id INTEGER REFERENCES test_parent(id),
11     description TEXT,
12     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
13 );\ "
14 su - postgres -c "psql -p 9193 -c \"
15 INSERT INTO test_parent (name) VALUES
16     ('Parent 1'),
17     ('Parent 2'),
18     ('Parent 3');
19
20 INSERT INTO test_child (parent_id, description) VALUES
21     (1, 'Child of Parent 1'),
22     (2, 'Child of Parent 2'),
23     (3, 'Child of Parent 3');\ "

```

Зафиксировал время перед добавлением новых данных:

Листинг 29: kitty

```

1 BEFORE_INSERT_TIME="2025-04-03 21:30:15"

```

Добавил новые записи в таблицы:

Листинг 30: kitty

```

1 su - postgres -c "psql -p 9193 -c \"
2 INSERT INTO test_parent (name) VALUES
3     ('New Parent 1'),
4     ('New Parent 2'),
5     ('New Parent 3');
6
7 INSERT INTO test_child (parent_id, description) VALUES
8     (4, 'Child of New Parent 1'),
9     (5, 'Child of New Parent 2'),
10    (6, 'Child of New Parent 3');\ "

```

Переключил WAL для обеспечения архивирования:

Листинг 31: kitty

```

1 su - postgres -c "psql -p 9193 -c \"SELECT pg_switch_wal();\ "

```

4. Симуляция логического повреждения данных

Зафиксировал время перед внесением ошибки:

Листинг 32: kitty

```

1 ERROR_TIME="2025-04-03 21:35:25"

```

Посмотрел текущие данные и выполнил некорректное обновление внешних ключей:

Листинг 33: kitty

```

1 su - postgres -c "psql -p 9193 -c \"
2 UPDATE test_child
3 SET parent_id = (
4     SELECT ceil(random() * 1000)::int
5 )
6 WHERE parent_id IS NOT NULL;
7
8 SELECT * FROM test_child LIMIT 10;\ "

```

После этого снова переключил WAL для архивирования:

Листинг 34: kitty

```
1 su - postgres -c "psql -p 9193 -c \"SELECT pg_switch_wal();\""
```

5. Восстановление данных до момента перед ошибкой

Остановил PostgreSQL:

Листинг 35: kitty

```
1 su - postgres -c "pg_ctl stop -D /var/lib/postgresql/data -m fast"
```

Скопировал базовую резервную копию в директорию данных:

Листинг 36: kitty

```
1 rm -rf /var/lib/postgresql/data/*
2 cp -a /tmp/pg_basebackup/* /var/lib/postgresql/data/
3 chown -R postgres:postgres /var/lib/postgresql/data
```

““

Настроил конфигурацию восстановления для PostgreSQL 17:

Листинг 37: kitty

```
1 cat > /var/lib/postgresql/data/postgresql.conf << EOF
```

Параметры восстановления

Листинг 38: kitty

```
1 restore_command = 'cp /var/lib/postgresql/wal_archive/%f %p'
2 recovery_target_time = '2025-04-03 21:35:25'
3 recovery_target_action = 'pause'
4 EOF
5
6 touch /var/lib/postgresql/data/recovery.signal
7 chown postgres:postgres /var/lib/postgresql/data/recovery.signal
```

Запустил PostgreSQL в режиме восстановления:

Листинг 39: kitty

```
1 su - postgres -c "pg_ctl start -D /var/lib/postgresql/data -o '-p 9193'"
```

Проверил статус восстановления и продолжил восстановление:

Листинг 40: kitty

```
1 su - postgres -c "psql -p 9193 -c \"SELECT pg_is_in_recovery();\""
2 su - postgres -c "psql -p 9193 -c \"SELECT pg_wal_replay_resume();\""
```

После завершения восстановления удалил recovery.signal и перезапустил PostgreSQL в обычном режиме:

Листинг 41: kitty

```
1 rm -f /var/lib/postgresql/data/recovery.signal
2 su - postgres -c "pg_ctl restart -D /var/lib/postgresql/data -o '-p 9193'"
```

6. Проверка результатов восстановления

Проверил данные в таблице после восстановления:

Листинг 42: kitty

```
1 su - postgres -c "psql -p 9193 -c \"SELECT * FROM test_child LIMIT 10;\""
```