Практическая работа №5.
Репликация и шардирование

Цель работы: научиться пользоваться базовыми инструментами репликации данных, встроенных в СУБД Postgres.

Теоретический материал

Репликация — это процесс копирования данных с одного сервера (основного, primary) на один или несколько других серверов (реплик, standby). Основная цель репликации — обеспечить отказоустойчивость и повысить доступность данных.

Типы репликации в PostgreSQL:

1. *Синхронная репликация*:
   - Основной сервер ждет подтверждения от реплики перед завершением транзакции.
   - Гарантирует полную согласованность данных, но может снизить производительность.
2. *Асинхронная репликация*:
   - Основной сервер не ждет подтверждения от реплики.
   - Обеспечивает более высокую производительность, но возможна небольшая задержка в синхронизации данных.
3. *Логическая репликация*:
   - Реплицируются только изменения данных (на уровне строк), а не весь журнал транзакций.
   - Позволяет реплицировать отдельные таблицы или даже части таблиц.

Для чего делается репликация:

- Отказоустойчивость: При сбое основного сервера реплика может взять на себя его роль.
- Распределение нагрузки: Реплики могут использоваться для выполнения запросов на чтение, разгружая основной сервер.
- Географическое распределение: Данные можно реплицировать в разные регионы для снижения задержек.

Шардирование — это метод горизонтального масштабирования, при котором данные распределяются между несколькими серверами (шардами). Каждый шард содержит часть данных, и вместе они образуют логически единую базу данных. Шардирование работает следующим образом:

1. Данные разделяются по определенному правилу (например, по диапазону значений или хэшу).
2. Каждый шард отвечает за свою часть данных.
3. Запросы распределяются между шардами в зависимости от того, где находятся нужные данные.

Для чего делается шардирование:

- **Горизонтальное масштабирование**: Возможность обрабатывать большие объемы данных, добавляя новые шарды.
- **Повышение производительности**: Нагрузка распределяется между несколькими серверами.
- **Гибкость**: Можно использовать разные стратегии шардирования в зависимости от требований.

Репликация и шардирование могут использоваться вместе для создания высокодоступных и масштабируемых систем. Например:

- Каждый шард может быть реплицирован для обеспечения отказоустойчивости.
- Реплики шардов могут использоваться для выполнения запросов на чтение, что повышает общую производительность системы.

Практическая часть

1. Создание нескольких серверов для репликации

Создайте на машине сервер, который будет использоваться в качестве основного для репликации. Подробно это разбиралось в практической работе №4. Можно добавить, что иногда порт, прописанный в конфигурационном файле, не подхватывается службой Windows при старте. В этом случае службу необходимо остановить, выполнив команду:

```
net stop <имя_службы>
```

Затем службу необходимо удалить, выполнив команду:

```
pg_ctl unregister -N <имя_службы>
```

Ещё службу можно удалить следующей командой (как вариант):

```
sc delete <имя_службы>
```

После того, как служба удалена, можно заново её зарегистрировать, явно указав порт в команде регистрации службы:

```
pg_ctl register -N <имя_службы> -D <каталог_данных>
      -o "-p <порт>"
```

Например:

```
pg_ctl register -N "postgresql-x64-15" -D
      "C:\Program Files\PostgreSQL\15\data" -o "-p 5433"
```

После этого зарегистрированная служба будет работать на указанном при регистрации порте и игнорировать порт, прописанный в конфигурационном файле. В принципе, данный способ регистрации служб является вспомогательной мерой, использовать его без необходимости не рекомендуется.

2. Настройка основного сервера

1. На основном сервере разверните базу данных из курсового проекта по курсу «Управление данными». В крайнем случае можно использовать учебную базу данных CanteenDishes.

2. Создайте директорию для хранения архивных копий журнала транзакций сервера.

3. Откройте конфигурационный файл основного сервера **postgresql.conf** и пропишите в нём следующие параметры:

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 1GB
archive_mode = on
archive_command = 'copy "%p" "C:\\path\\to\\archive\\%f"'
```

wal_level устанавливает уровень детальности протоколирования.

max_wal_senders устанавливает количество процессов операционной системы, которые могут заниматься оправкой данных

wal_keep_size – объём файлов с журналом транзакций, хранимых на основном сервере

archive_command – команда операционной системы для сохранения архивов журнала транзакций. Архивы журналов транзакций должны сохраняться в папку, созданную на шаге 2.

4. Перезапустите службу, подключитесь к серверу в pgAdmin и в Query Tool введите команду для создания роли репликатора:

```
CREATE ROLE replicator WITH REPLICATION LOGIN PASSWORD
'ваш_пароль';
```

Чтобы команда прошла успешна, у пользователя, под которым она выполняется, должны быть права суперпользователя.

5. В файле pg_hba.conf в директории, где развёрнут сервер, добавьте строку:

```
host  replication  replica_user  127.0.0.1/32 scram-sha-256
```

Имя пользователя, которое записывается в третью колонку, должно совпадать с тем, которое было указано при создании роли на шаге 4. В качестве метода аутентификации можно использовать не только **scram-sha-256**, но и **md5**.

Снова перезапустите сервер.

3. Настройка сервера-реплики

1. Создайте пустую папку, в которой будет находиться реплика.

2. Выполните в командной строке Windows следующую команду:

```
pg_basebackup -h <IP> -p <port> -U <user>
      -D "<path>" -P -R
```

где

IP – IP-адрес текущей машины (можно 127.0.0.1);

port – порт, на котором работает основной сервер;

user – пользователь, созданный для репликации во время настройки сервера;

path – путь к папке, созданной на шаге 1.

В результате создастся новый каталог данных, являющийся репликой существующего каталога.

3. Войдите в конфигурационный файл postgresql.conf и добавьте в него следующую информацию:

```
hot_standby = on
port = XXX
```

В качестве порта необходимо задать незанятый порт.

4. Зарегистрируйте службу Windows для нового сервера, выполнив следующую команду:

```
pg_ctl register -N <имя_службы> -D <каталог_данных> -S auto
```

5. Создайте подключение к созданной службе и убедитесь, что подключение проходит успешно.

4. Проверка работы репликации

1. На основном сервере выполните команду

```
SELECT * FROM pg_stat_replication;
```

В результате должна появиться информация о подключенных серверах.

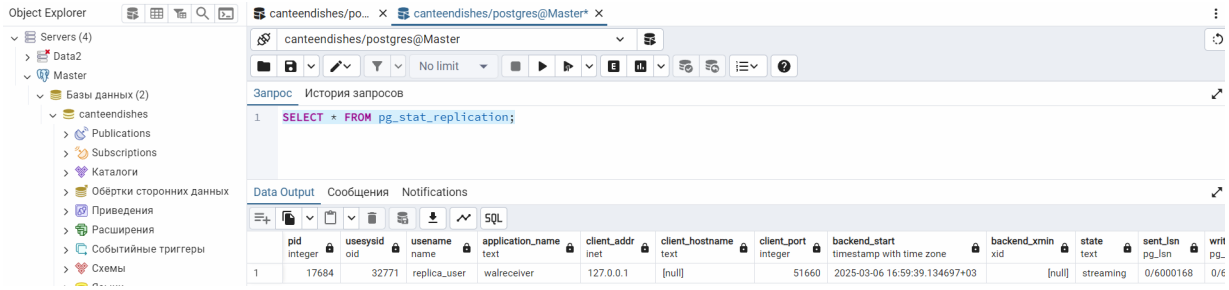

Рисунок 1. Информация о подключенных репликах на основном сервере

2. На сервере реплики выполните команду:

```
SELECT pg_is_in_recovery();
```

В результате должна вывестись истина, свидетельствующая о том, что текущий сервер является репликой, и проходит репликация.

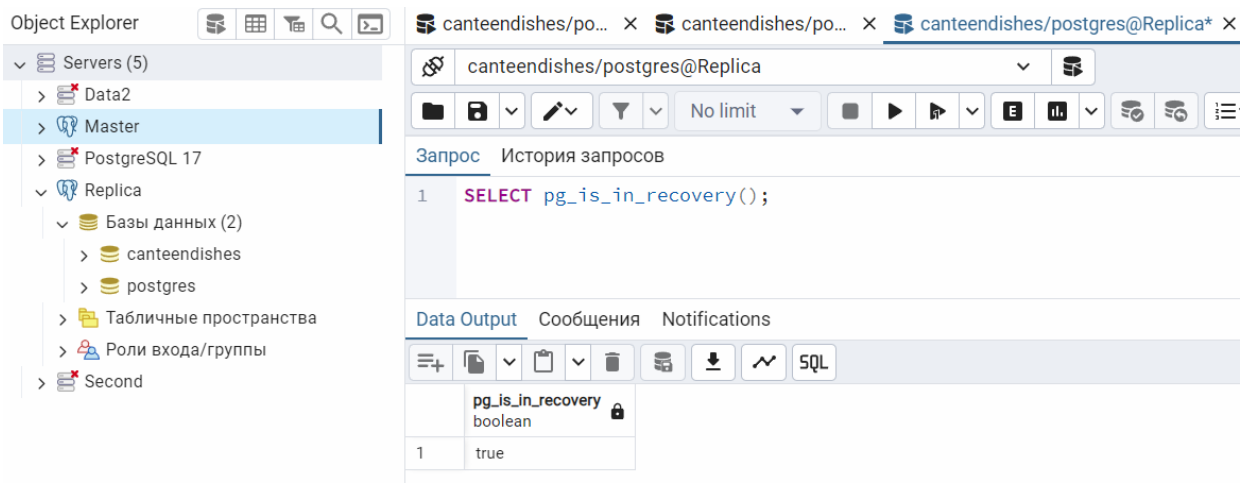

Рисунок 2. Информация о включенной репликации на реплике

3. Внесите изменения в данные, хранящиеся на основном сервере. Убедитесь в том, что эти данные были изменены на реплике.

5. Настройка шардирования

1. Установите программу Citus, скачав с официального сайта: https://www.citusdata.com/download/

2. Создайте сервера для координатора и для рабочего сервера, как описано в разделе 1, работающие на разных портах.

3. В конфигурационных файлах обоих серверов postgresql.conf пропишите следующий параметр:

```
shared_preload_libraries = 'citus'
```

4. Зарегистрируйте службы для серверов и запустите их

5. Подключитесь к серверу-координатору и выполните следующие команды:

```
CREATE EXTENSION citus;
SELECT * from master_add_node('<хост>', <порт>);
SELECT * FROM master_get_active_worker_nodes();
```

Первая команда активирует Citus на сервере-координаторе. Вторая команда подключит к нему второй сервер как шард. Третья команда выведет информацию об успешно подключенном шарде.

6. Создайте таблицу на координаторе и выполните команду:

```
SELECT create_distributed_table('<таблица>', '<колонка>');
```

В качестве таблицы в команде укажите имя созданной таблицы, в качестве колонки – имя её колонки, по которой должно осуществляться шардирование.

7. Вставьте данные в таблицу и выполните запрос на чтение из таблицы. Убедитесь, что данные, вставленные в таблицу, были успешно прочитаны.

8. Подключитесь к рабочему серверу. Убедитесь, что там создалось множество таблиц, каждая из которых содержит часть данных, вставленных в таблицу на сервере-координаторе.

9. Выполните запрос на получение данных из координатора, перед словом **SELECT** написав слово **EXPLAIN**. Убедитесь, что в реальности данные собираются из шардов, что отражается в плане выполнения запроса.

6. Особенности работы с Citus в Windows

Готовой версии Citus под Windows нет. В связи с этим, для установки Citus в Windows необходимо использовать Docker. Порядок работы:

1. Скачать Docker по ссылке https://docs.docker.com/desktop/setup/install/windows-install/
2. Запустить **docker engine**, чтобы он работал в фоне

3. Через консоль создать сервер с помощью команды:

```
docker run -d --name <name> -p <port>:5432 -e
POSTGRES_PASSWORD=<pass> citusdata/citus:13.0
```

где **<port>** - порт, к которому мы будем подключаться в pgAdmin

4. Соедините созданные сервера в сеть, чтобы можно было к ним обращаться по названию:

```
docker network create <network-name>
docker network connect <network-name> <worker-name>
```

5. В pgAdmin создаем сервер с указанием порта, который использовали в команде создания сервера.
6. Работающие сервера можно проверить через команду **docker ps** в консоли. Чтобы убедиться, что созданные сервера находятся в одной сети, выполните команду **docker network inspect citus**. Должна вывестись конфигурация JSON, содержащая созданные сервера, как показано на рисунке 3. Красным выделены имена созданных серверов, введённые соответствующими командами.

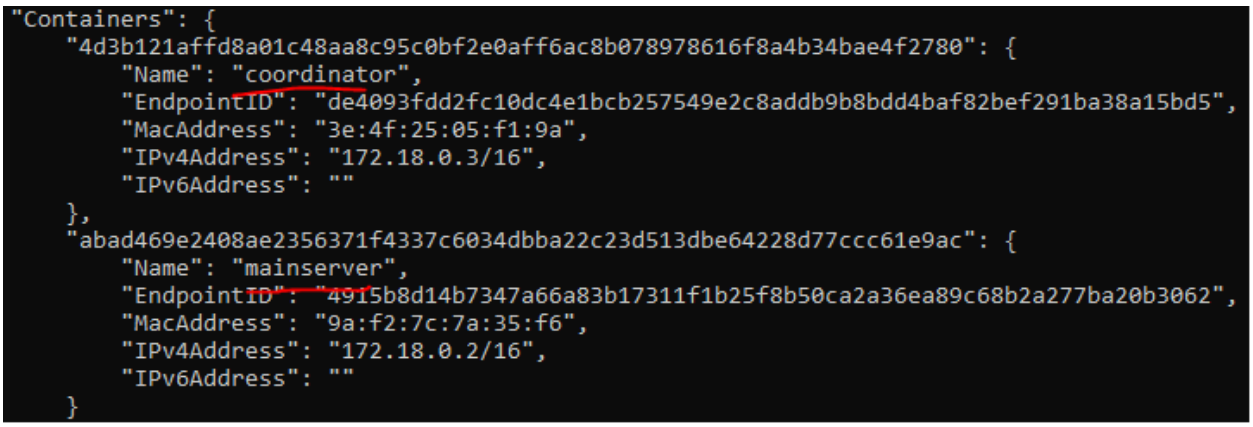

Рисунок 3. Фрагмент конфигурации Docker после выполнения команды **docker network inspect citus**.

Иногда Docker не подхватывает пароли для пользователя PostgreSQL. Для исправления этой проблемы необходимо в командной строке в файл pgpass записать:

```
docker exec -it coordinator bash
echo "mainserver:5432:postgres:pass" > ~/.pgpass
chmod 600 ~/.pgpass
```

Задание

1. Создайте основной сервер для репликации
2. Разверните на нём базу данных из курсового проекта, либо, в крайнем случае, учебную базу данных CanteenDishes
3. Настройте репликацию основного сервера, где развёрнута база данных
4. Создайте реплику этого сервера и службу для неё на другом порту
5. Запустите реплику и убедитесь в том, что репликация работает, внеся изменения в данные на основном сервере.
6. Установите Citus
7. Создайте ещё один дополнительный сервер для шардирования, работающий на третьем порту
8. Подключите к основному серверу дополнительный сервер в качестве рабочего
9. Создайте на основном сервере новую таблицу и включите для неё шардирование
10. Проверьте работу шардирования, внеся данные и посмотрев, где

Оформление отчёта

Отчёт должен иметь титульный лист с указанием ФИО и группы студента, а так же темы практической работы. После этого должен быть указан порт и название службы основного сервера, с которым будет вестись работа (скриншот командной строки). Отчёт должен содержать три результата выполнения запроса к таблице **`pg_stat_replication`**: до начала настройки репликации, после создания пользователя и резервной копии для репликации, после включения сервера-реплики. Кроме того, там должны быть скриншоты созданием службы для сервера-реплики и с двумя проверками работы репликации: через вызов функции **`pg_is_in_recovery()`** на сервере-реплике и через редактирование данных на основном сервере (что хранится в реплике до редактирования данных в основной БД и после). Далее в отчёте должен быть скрипт создания таблицы для включения шардирования и результаты выполнения простого запроса к ней на чтение (данные в таблицу при этом нужно внести) и с планом выполнения запроса. Кроме того, нужно привести список созданных шардов.