Задание №3: Репликация

# Настройка асинхронной репликации

Перед запуском контейнеров с PostgreSQL были подготовлены два конфигурационных файла:

* postgresql.conf
* pg\_hba.conf

В файле postgresql.conf включены следующие настройки:

wal\_level = replica

max\_wal\_senders = 3   
max\_replication\_slots = 3

В файл pg\_hba.conf добавлена строка:

host    replication     all             172.19.0.0/16           md5

Данные файлы были «проброшены» в контейнер мастера.

В файл инициализации контейнера init.sh добавлены скрипты для копирования подготовленных файлов, создания пользователя для репликации и выдачи ему нужных прав:

cat /tmp/postgresql.conf > /var/lib/postgresql/data/postgresql.conf   
cat /tmp/pg\_hba.conf > /var/lib/postgresql/data/pg\_hba.conf

psql -v ON\_ERROR\_STOP=1 --username "$POSTGRES\_USER" --dbname "$POSTGRES\_DB" <<-EOSQL   
    CREATE ROLE replication\_user WITH LOGIN PASSWORD 'PASSWORD' REPLICATION;   
    GRANT CONNECT ON DATABASE otus\_social\_network TO replication\_user;   
EOSQL   
  
psql -v ON\_ERROR\_STOP=1 --username "$POSTGRES\_USER" --dbname "otus\_social\_network" <<-EOSQL   
    GRANT SELECT ON ALL TABLES IN SCHEMA public TO replication\_user;   
    GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO replication\_user;   
    GRANT USAGE ON SCHEMA public TO replication\_user;   
    ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO replication\_user;   
EOSQL

Так же был создан контейнер для реплики БД.

После запуска контейнеров СУБД и приложения, выполнена команда создания резервной копии узла:

docker exec -ti postgres\_db\_master su postgres -c "mkdir /var/lib/postgresql/data/pgslave"

docker exec -ti postgres\_db\_master su postgres -c "pg\_basebackup --username=replication\_user --pgdata=/var/lib/postgresql/data/pgslave --wal-method=stream --write-recovery-conf"

После чего контейнер реплики был остановлен и файлы pgdata были заменены только что сделанным бэкапом и контейнер снова был запущен.

В файле postgresql.auto.conf были обновлены настройки подключения к мастеру:

primary\_conninfo = 'user=replication\_user host=postgres\_db\_master port=5432 password=PASSWORD application\_name=walreceiver channel\_binding=prefer sslmode=prefer sslcompression=0 sslsni=1 ssl\_min\_protocol\_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres target\_session\_attrs=any'

# Переключение запросов с мастера на реплику

В код приложения были внесены необходимые изменения для переключения запросов [/user/get/{id}](https://github.com/OtusTeam/highload/blob/master/homework/openapi.json#L129) и [/user/search](https://github.com/OtusTeam/highload/blob/master/homework/openapi.json#L165) на реплику и проведены необходимые тесты.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рис1. Нагрузка на мастер до переключения приложения на реплику при 1000 потоках на /user/search

Изображение выглядит как снимок экрана, текст, диаграмма, линия

Автоматически созданное описание

Рис.2 Нагрузка на мастер после переключения приложения на реплику при 1000 потоках на /user/search

Изображение выглядит как снимок экрана, текст, диаграмма

Автоматически созданное описание

Рис.3 Нагрузка на реплику после переключения приложения на реплику при 1000 потоках на /user/search

# Настройка второго slave синхронной репликации

Slave для синхронной репликации создавался аналогично с первым, кроме:

В файле postgresql.conf мастера была сделана настройка:

synchronous\_commit = on

synchronous\_standby\_names = 'FIRST 1 (walreceiversync, walreceiver)'

В файле postgresql.auto.conf были обновлены настройки подключения к мастеру:

primary\_conninfo = 'user=replication\_user host=postgres\_db\_master port=5432 password=PASSWORD application\_name=walreceiversync channel\_binding=prefer sslmode=prefer sslcompression=0 sslsni=1 ssl\_min\_protocol\_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres target\_session\_attrs=any'

# Нагрузка на запись

Нагрузка на запись создавал путем запуска скрипта в LinqPad в 2 потока:

void Main()   
{   
    var countBefore = User.Count();   
    Console.WriteLine($"Row count before {countBefore}");   
       
    var testCount = 10000;   
    var i = 1;   
    for(; i <= testCount; i++)   
    {   
        var user = new User   
        {   
            PublicId = System.Guid.NewGuid(),   
            Firstname = "John",   
            Secondname = $"Doe {i}",   
            BirthDate = DateOnly.FromDateTime(DateTime.Now),   
            City = "SimCity",   
            Biography = "Fake",   
            PasswordHash = "fake\_hash"   
        };   
  
        try   
        {   
            User.Add(user);   
  
            SaveChanges();   
               
            Console.WriteLine($"{i} saved");   
        }   
        catch (Exception ex)   
        {   
            Console.WriteLine(ex.Message);   
            break;   
        }   
    }   
       
       
    Console.WriteLine($"Max row count after {countBefore + i}");   
}

Через некоторое время после запуска контейнер мастера был «убит»:

docker kill --signal=SIGKILL postgres\_db\_master

При проверке выяснилось что в реплики все данные сохранились без потерь.

# Promote одной из реплик

Подключился к синхронной реплике:

docker exec -it postgres\_db\_slave\_2 su - postgres -c psql

И выполнил команду:

select \* from pg\_promote();

После изменил в файле postgresql.conf параметры:

synchronous\_commit = on   
synchronous\_standby\_names = 'ANY 1 (postgres\_db\_master, walreceiver)'

И выполнил команду:

select \* from pg\_reload\_conf();