**О Т Ч Е Т**

о выполнении долгосрочного домашнего задания

по учебной дисциплине: «Основы построения защищенных баз данных»

Выполнил: Шараев Ц.А

**Поликлиника.**

1)-4) выполнены в прошлом ДДЗ.

5)Ролевая модель безопасности БД.

* admin(полный доступ на все)
* maindoctor (чтение всех + изменение и удаление таблицы doctor)
* patient(только чтение)

Скрипты создания ролей и назначения привелегий в файле my.txt.

6)RLS

Созданы политики для таблицы patient:

otherRLS(все строки)

doctorRLS(получает только тех пациентов, которых лечит)

patientRLS(получает только свою строку)

Скрипты создания политик в файле my.txt.

7)В PostgreSQL аудит событий уровня базы данных, таблиц и уровня строк можно реализовать при помощи параметра log\_statement = 'all' в файле C:\Program Files\PostgreSQL\15\data\postgresql.conf. Теперь любое взаимодействие с базой данных будет записываться в файлы формата txt по пути C:\Program Files\PostgreSQL\15\data\log.

Создан журнал аудита log\_ddl, в который записываются все события, связанные с выполнением DDL команд.

Скрипты создания в файле my.txt.

8) В PostgreSQL аудит событий уровня таблиц можно реализовать совместно с аудитом событий уровня строк. Создана таблица log, которая является журналом аудита и создана одна триггерная функция для всех триггеров insert\_log().

Далее на каждую таблицу создано два триггера, один на операции CUD, другой на операцию TRUNCATE( audit\_tables, audit\_tables\_truncate).

Далее для успешной работы логирования событий назначается всем существующим ролям право на запись в таблицу log.

Скрипт создания в файле my.txt.

9) Проверка правил (ограничений) предметной области с помощью декларативных CONSTRAINT средств контроля целостности реализована в пункте 4 ДДЗ при создании таблиц.

Проверка правил (ограничений) предметной области с помощью процедурных (триггеры уровня строки/таблицы) средств контроля целостности реализована с помощью триггерной функции ch\_cap().

Скрипт создания в файле my.txt.

10)Шифрование данных

Шифрование данных реализовано на примере таблицы cred, в которой защищался столбец password.

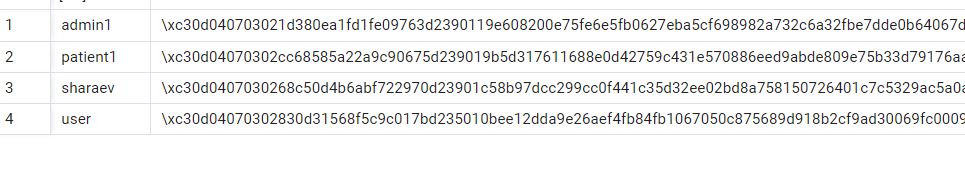
Для каждого пользователя были сгенерированы ключи шифрования, которые хранятся в отдельной таблице базы данных user\_key.

Реализована триггерная функция шифрования данных при добавлении/изменении, которая использует алгоритм шифрования данных по умолчанию (AES 128) на ключе пользователя, который вызвал операцию.(change\_encrypt\_clients)

Создана вспомогательная таблица, в которой будет указываться кто последний зашифровал конкретную строку в таблице cred. (last\_change)

Для созданных таблиц были реализованы политики RLS для обеспечения контроля доступа к ним.

Чтобы при выборке данные были расшифрованы на ключе пользователя, запросившего данную операцию, написана функция(decrypt\_clients).

11) Репликация

Была выбрана политика потоковой асинхронной репликации в режиме “master-slave”.

Primary-сервер:192.168.3.1 (win10)

Standby-сервер:192.168.3.129 (win10)

На primary-сервере (master) необходимо разрешить пользователю подключаться из standby-сервера(slave) . Для этого в pg\_hba.conf добавляем запись :

host replication postgres 192.168.3.129 scram-sha-256

Далее редактируем следюущие параметры в postgresql.conf(при необхожимости снимаем комментарии):

listen\_addresses = 'localhost, 192.168.3.1'

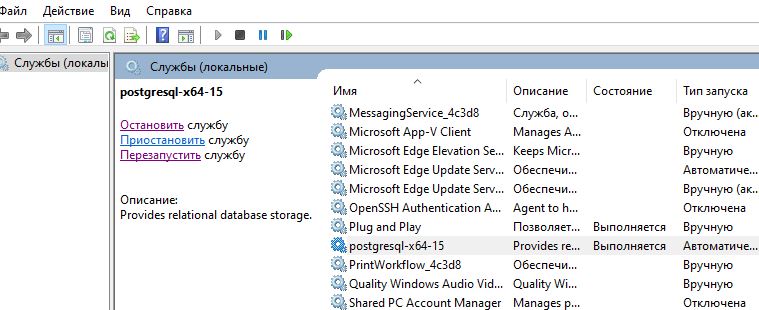
wal\_level = hot\_standby

archive\_mode = on

archive\_command = 'cd .'

max\_wal\_senders = 2

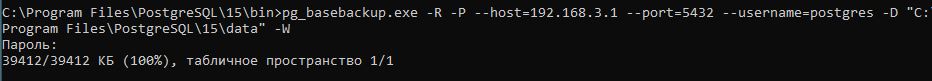
Перезапускаем службу postgresql:



Переходим к настройке резервного сервера:

Останавливаем службу Postgresql.

Удалем все сожержимое директории data.

Далее вводим команды, показанне на рисунке:

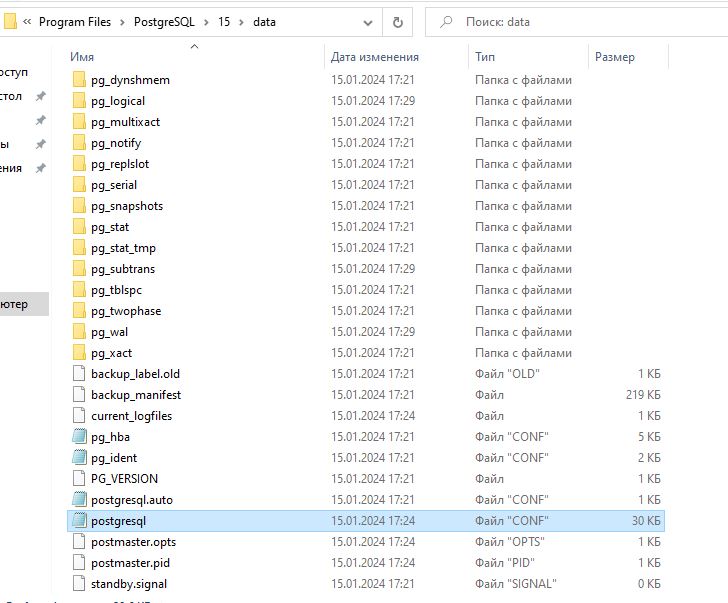
Затем редактируем файл postgresql.conf:

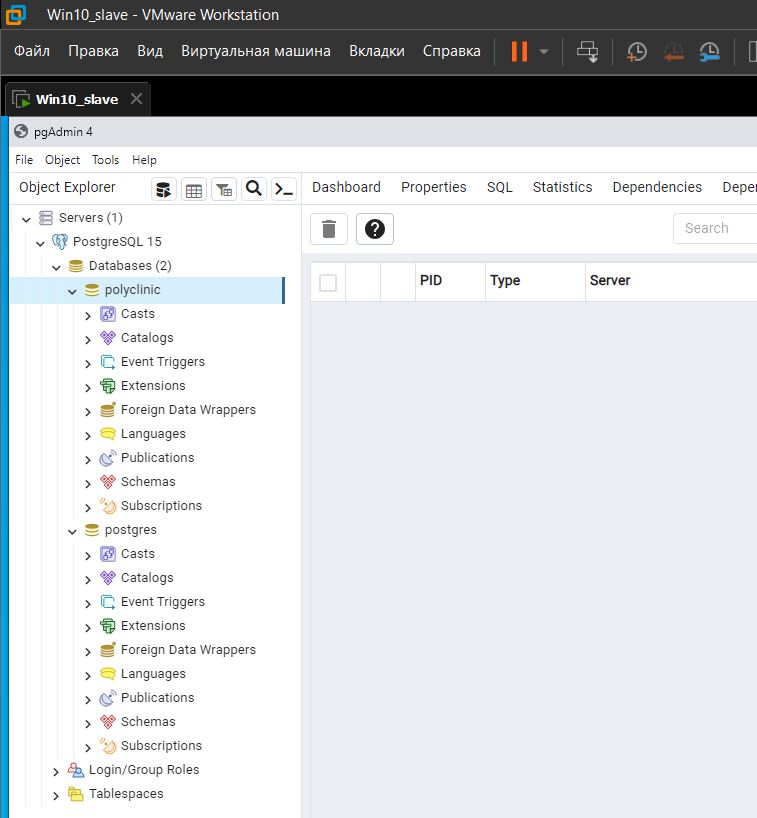
standby\_mode = 'on'

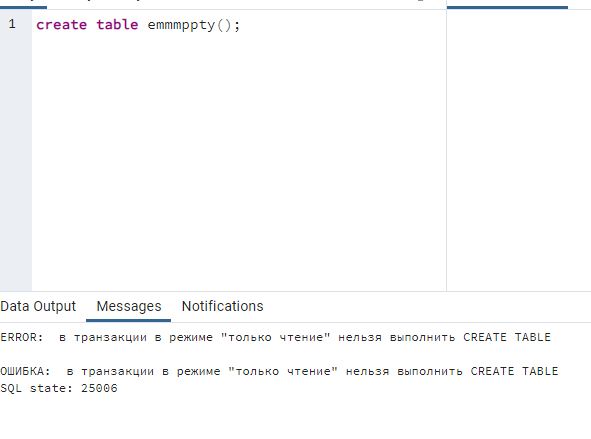
primary\_conninfo = 'host=192.168.3.1 port=5432 user=postgres password=root'

hot\_standby = on

Запускаем службу Postgresql.

Файл standby.signal является признаком репликации.

Проверям работу репликации:

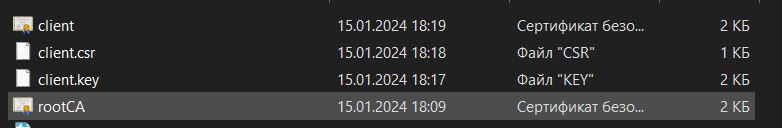
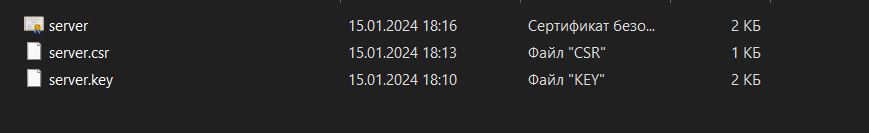
12-13) Внешняя аутентификация

PostgreSQL поддерживает SSL-соединения, которые позволяют пользователям безопасно подключаться к своим базам данных.

На сервере PostgreSQL нам нужны 3 сертификата (например, в каталоге данных) для настройки SSL:

‐ rootCA.crt (доверенный корневой сертификат)

‐ server.crt (сертификат сервера)

‐ server.key (закрытый ключ)

openssl genrsa -out rootCA.key 2048

openssl req -x509 -new -key rootCA.key -days 10000 -out rootCA.crt

Аналогично, сгенерируем ключ для сервера server.key:

openssl genrsa -out server.key 2048

Теперь создаем запрос на сертификат. При заполнении полей, в поле Common Name важно указать имя сервера: домен или IP адрес. (server.csr):

openssl req -new -key server.key -out server.csr

Подписываем запрос на сертификат нашим корневым сертификатом. (server.crt):

openssl x509 -req -in server.csr -CA ../rootCA.crt -CAkey ../rootCA.key -CAcreateserial -out server.crt -days 5000

Теперь проделываем эти три последние команды снова, но уже создаем связку ключ-сертификат для потенциального клиента. Имена файлов соответственно меняем (например client.key, client.crt, client.csr). В поле Common Name указываем логин под которым будет выполняться подключение к БД:

openssl genrsa -out client.key 2048

openssl req -new -key client.key -out client.csr

openssl x509 -req -in client.csr -CA ../rootCA.crt -CAkey ../rootCA.key -CAcreateserial -out client.crt -days 5000

Копируем корневой сертификат CA, ключ и сертификат сервера в каталог БД (data)

Обновим файл pg\_hba.conf, чтобы добавить следующие строки:

hostssl all all 192.168.3.129/32 trust clientcert=verify-full

Также обновим файл postgresql.conf:



Перезапусакем службу postgresql.

Далее передаем client.crt, client.key и rootCA.crt на клиент(Win10 с ip: 192.168.3.129)

Результат:

