**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет ИТМО

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа 4. Маскировка и анонимизация данных**

По дисциплине «Проектирование баз данных»

Выполнил:

студент группы №М3212

*Тимофеев Вячеслав*

Проверила:

*Чеботарева*

Санкт-Петербург

2025

**Задачи:**

1. Замаскировать поля с конфиденциальными данными.

2. Провести анонимизацию данных.

**Ход работы:**

1. tl;dr: Для работы с PostgreSQL Anonymizer установил расширение

( <https://gitlab.com/dalibo/postgresql_anonymizer.git> ) в ядре Linux-контейнера Docker.

Структура контейнера:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Контент, сгенерированный ИИ, может содержать ошибки.

Dockerfile

Устанавливаем зависимости для сборки расширений:



Клонируем репозиторий PostgreSQL Anonymizer от Neon Database:

****

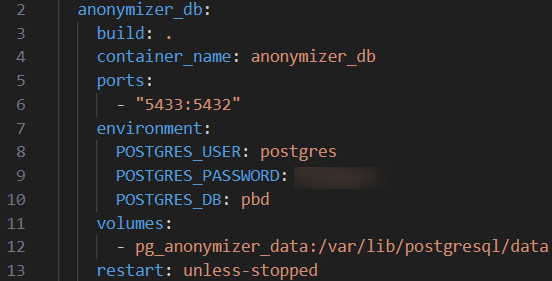
docker-compose.yml

Билдим контейнер в текущей дирекотрии (build: . ), заведомо закинув в неё Dockerfile;

Привязываем локальный порт 5433 с Windows на локальный порт 5432 поднятого внутри контейнера Linux + PostgreSQL;

Указываем volumes для хранения данных контейнера на хостовой машине (чтобы избежать потери при пересоздании контейнера);

Рестартим контейнер всегда при запуске, остановка при явном указании (restart: unless-stopped).

****

Билдим контейнер:

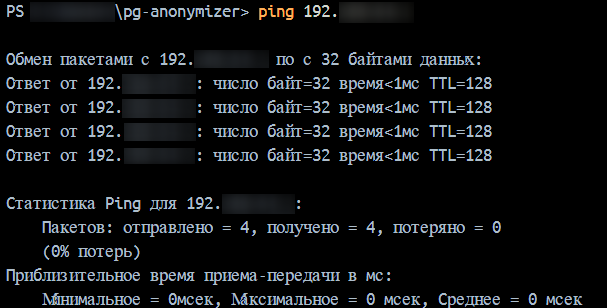
****

(Далее для запуска: docker compose up -d )

Проверяем, что запросы перенаправляются внутрь контейнера на корретный порт (с PostgreSQL):



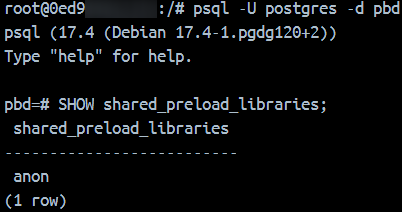
Проверяем обмен пакетов между контейнером и хостом:



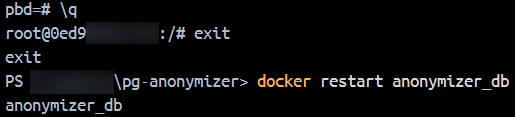
Заходим внутрь контейнера и устанавливаем расширение **anon** в загруженные на этапе запуска сервера:



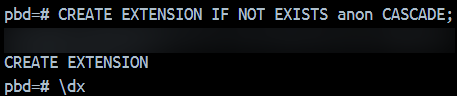
Подключаемся к базе с помощью psql (без явного пароля, так как на хосте в **pg\_hba.conf** вход по методу **scram-sha-256**) и проверяем наличие расширения в списке предварительных:

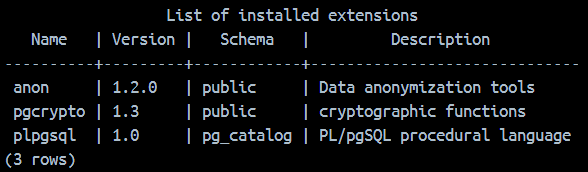


Перезагружаем контейнер:

****

Устанавливаем расширение и проверяем его в списке установленных:



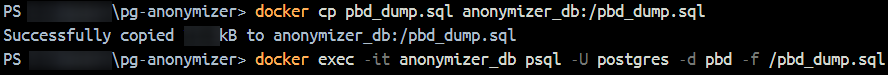
****

(**pgcrypto** – вспомогательное расширение для работы функций генерирования случайных значений и хеширования)

Делаем дамп локальной базы данных, поднятой на Windows:



Копируем дамп внутрь контейнера и восстанавливаем там:

****

Создаю новый сервер в pgAdmin для работы с поднятной в контейнере бд (с портом 5433, через который пробрасываем запросы с Windows) с ролью **суперюзера** (postgres)):

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Проверим наличие расширения:

Изображение выглядит как текст, снимок экрана, Шрифт, число

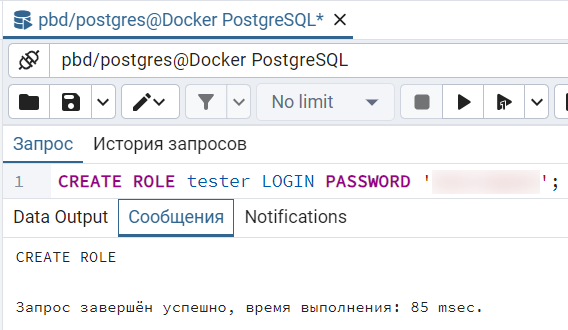
Контент, сгенерированный ИИ, может содержать ошибки.

Инициализируем служебную схему и функции для работы расширения:

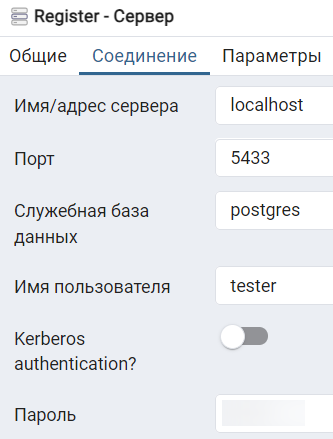
Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

1. Создадим новую роль без прав суперюзера для тестирования маскировки данных (суперюзер видит данные не замаскированными, даже если включено маскирование):



Создаю новый сервер в pgAdmin для работы с поднятной в контейнере бд с ролью не суперюзера (tester):



Теперь имеем два сервера: один для модификаций и один для тестирования:

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Контент, сгенерированный ИИ, может содержать ошибки.

**Выбранные поля для маскировки:**

user:

name, surname, patronymic, email, phone\_number, registration\_date, password

company:

Inn, ogrn, kpp

Создадим функцию для генерации рандомного пароля (длиной от 8 до 24 символов):

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Создадим метки безопасности на выбранные поля:

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Включим прозрачное маскирование через представления:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

В схеме **mask** (которая появиалсь при инициализации расширения) хранятся представления с установленными SECURITY LABEL:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, число, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Выдадим созданной роли права на использование схемы **mask**, уберем права на SELECT из оригинальных таблиц:

Изображение выглядит как текст, электроника, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Поменяем **search\_path** для роли не суперюзера так, чтобы запросы сначала шли в схему **mask**:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

(После чего нужно перезапустить локальную службу postgresql, либо просто перезайти в pgAdmin (или что используеутся), чтобы изменения search\_path вступили в силу)

Замаскированные данные от лица суперюзера:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Замаскированные данные от лица не суперюзера (tester):

Изображение выглядит как текст, число, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

1. Создадим материальное представление для таблиц **vacancy** (поля city, salary\_level) и **company** (поле industry), используя **Generalization**:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Данные из созданного представления **salary\_generalization**:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Применение – анализ распределения вакансий по городам, сфере и уровню зарплаты без точных чисел

Создадим материальное представление для таблиц **user** (поля email, name, surname), **resume** (поле title), **response** (поля id, response\_date) используя **Pseudonymization**:

Изображение выглядит как текст, электроника, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Данные из созданного представления **user\_responses\_pseudo**:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Применение – анализ откликов пользователей на вакансии без раскрытия персональных данных

Создадим материальное представление для таблиц **user** (поля id, registration\_date), **resume** (поля id, user\_id), **response** (поля id, response\_date, resume\_id), **chat** (поля id, user\_id) используя **Generic Hashing** и **Adding Noise**:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Данные из созданного представления **user\_behavior\_hash\_noise**:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Применение – анализ поведения пользователей в процессе поиска работы без раскрытия личности (сколько: откликов, чатов с работодателями, дней до первого отклика с момента регистрации)

В итоге имеем три созданных материальных представления в схеме **public**:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод:** в ходе лабораторной работы научился работать с расширением PostgreSQL Anonymizer. А именно: маскировать поля, проводить анонимизацию даных; создавать материальные представления, используя различные стратегии анонимизации. Убедился в корректности маскировки данных от лица обычного пользователя.