**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Проектирование баз данных**

**Лабораторная работа №4.**

Маскировка и анонимизация данных

Вариант №6 (Библиотека)

Выполнил студент группы №М34061

**Величко Максим**

Преподаватель:

**Боркунов Владимир Юрьевич**

САНКТ-ПЕТЕРБУРГ

2024

**Цель работы:**

В данной лабораторной работе была выполнена задача маскировки полей с конфиденциальными данными и анонимизации данных в базе данных PostgreSQL. Основной целью работы было применение различных методов анонимизации для защиты личной информации пользователей.

Изначально созданная мною база данных выглядела следующим образом:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Пример данных из таблицы users:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

У меня есть docker контейнер с Postgres, и использовал я **pg-anonymizer**

[**https://github.com/rap2hpoutre/pg-anonymizer**](https://github.com/rap2hpoutre/pg-anonymizer)

Перед этим в мой контейнер установил **npx**

**Маскировка данных**

Для маскировки конфиденциальных данных, таких как email и password, использовалась команда pg-anonymizer. Код для маскировки данных приведен ниже:

npx pg-anonymizer postgres://maksvell:maksvell@77.221.133.240:5432/db\_design --columns=email:faker.internet.email,password:faker.internet.password -o anonymized\_dump\_1.sql

И получил вот это:

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

**Создание Materialized Views**

С целью анонимизации данных были созданы три materialized view с использованием различных стратегий:

* **Generalization**: заменяет данные более широкими и менее точными значениями.
* **Adding Noise**: добавляет случайные значения к данным.
* **Randomization**: меняет значения данных на случайные.

Вот код:

-- Generalization View

CREATE MATERIALIZED VIEW mv\_generalized\_users AS

SELECT

id,

SUBSTRING(name FROM 1 FOR 1) || '\*\*\*\*\*' AS name, -- Generalization

email,

'\*\*\*\*\*' AS password,

registration\_date

FROM users;

-- Adding Noise View

CREATE MATERIALIZED VIEW mv\_noisy\_users AS

SELECT

id,

name,

email,

password,

registration\_date + (RANDOM() \* INTERVAL '30 days') -- Adding Noise

FROM users;

-- Randomization View

CREATE MATERIALIZED VIEW mv\_randomized\_users AS

SELECT

id,

name,

email,

password,

registration\_date + (INTERVAL '1 day' \* FLOOR(RANDOM() \* 365)) -- Randomization

FROM users;

Получил в итоге вот такие данные:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Далее подключаемся к Postgres** внутри контейнера

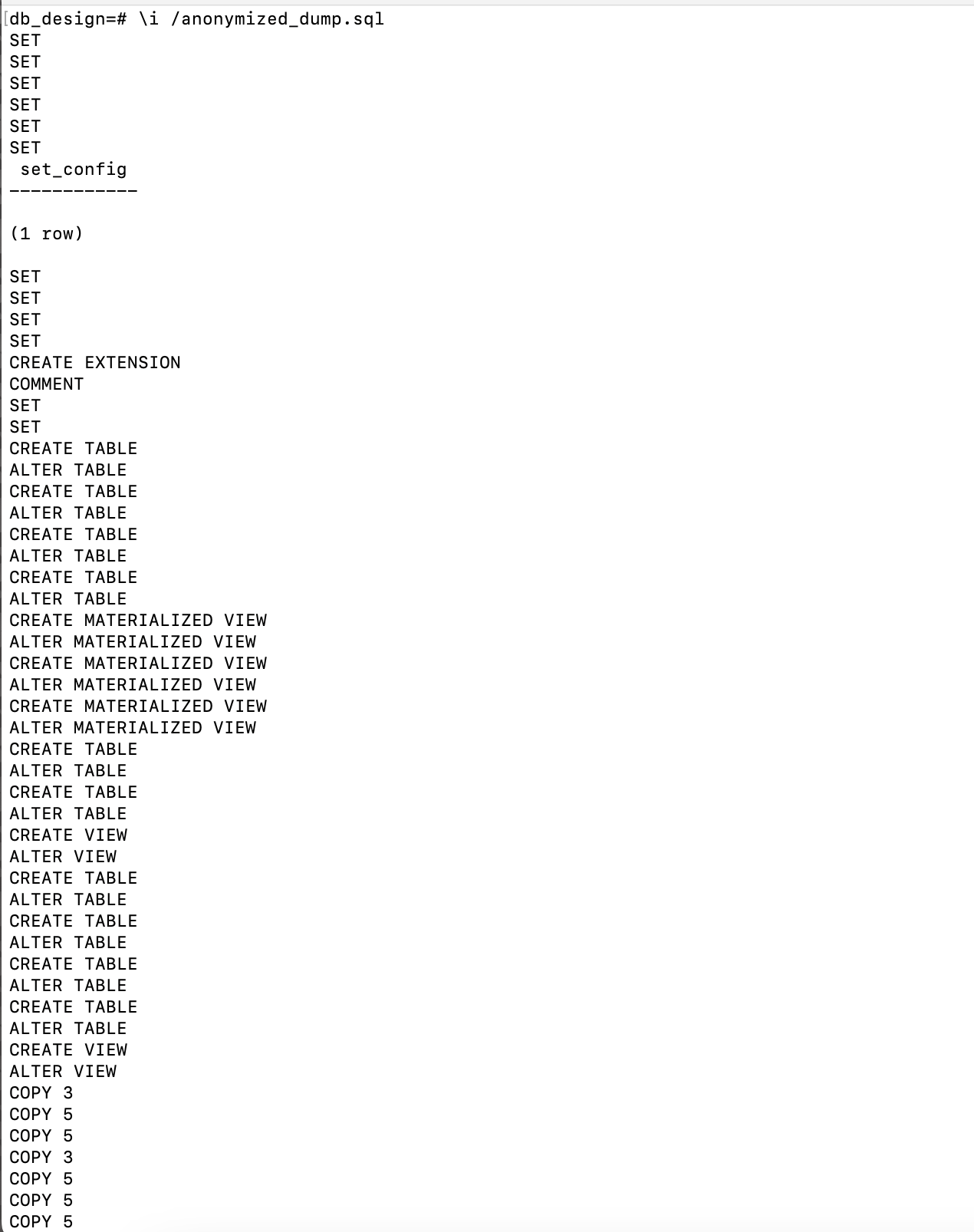
psql -U maksvell -d db\_design

**Перед этим обязательно удаляем все данные**

**Импортируем файл** anonymized\_dump\_1.sql.

\i /anonymized\_dump\_1.sql

И получаем вот это



Теперь смотрим изменения в самой бд

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Заключение**

В ходе лабораторной работы была успешно проведена маскировка и анонимизация конфиденциальных данных в базе данных PostgreSQL. Примененные стратегии анонимизации продемонстрировали эффективность защиты личной информации пользователей.