**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Информационная безопасность баз данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

«Реализация БД в рамках СУБД»

**Выполнили:**

Ахраров Али, студент группы N3350

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Салихов Максим Русланович

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

1. **Цель работы:**

Получение навыков созданию примитивных систем мониторинга, разграничения доступа и шифрования средствами СУБД.

1. **Задание**

Задачи по мониторингу БД:

* Создайте таблицу-лог, отдельную от ваших основных сущностей БД.
* Создайте для каждой основной таблицы в вашей БД триггер, который срабатывает при любых изменениях в БД (вставка новых данных, изменение существующих записей, удаление кортежей из таблицы). При срабатывании триггер должен вносить в таблицу-лог информацию о том, когда было произведено изменение, со стороны какой роли поступил запрос, какие кортежи поменялись, старые и новые значения.
* Продемонстрируйте работу системы логирования для различных операций и отношений.

Задачи по шифрованию данных.

* Создайте таблицу с секретными данными, отдельно от ваших основных сущностей. Например, это может быть таблица с токенами или ключами доступа, для каждого класса-пользователей.
* Зашифруйте содержимое данной таблицы, в качестве алгоритма шифрования используйте любой симметричный алгоритм шифрования. Ключ шифрования для данной таблицы не должен храниться в ИС. Ключ шифрования может быть получен из индивидуального пароля для дешифрования суперпользователя (пароль не связан с паролем для входа в СУБД). Индивидуальный пароль суперпользователя и ключ шифрования может быть связан через одностороннюю функцию. Например, пусть индивидуальный пароль комбинация «!stroNgpsw31234», считаем от данного пароля детерминированную хэш-функцию (например, sha-256), полученный хэш-используем как ключ шифрования/дешифрования для симметричного алгоритма шифрования таблицы с секретными данными (например, для AES-256)
* Демонстрируем, что даже обладая полными правами администратора, но без знания индивидуального пароля невозможно получить содержимое таблицы с секретными данными

Задачи по разграничению доступа в БД:

* Создайте в СУБД как минимум 2 роли (суперпользователь не считается) для каждого из классов потребителей информации;
* С помощью внутренних инструментов СУБД для каждой роли определите набор привилегий по отношению к таблицам вашей БД. Руководствуйтесь принципом минимальных привилегий, если определенному классу потребителей не нужен доступ к определенным таблицам/атрибутам (список задач БД, составленный в рамках 1 ЛР), то доступ к этим таблицам/атрибутам не предоставляется. Разграничиваем доступ к представлениям, созданным в 1 ЛР, а также таблицам логирования (таблицы логирования может просматривать только суперпользователь)
* Продемонстрируйте работу вашей системы разграничения доступа. Зайдите за каждую из ролей и покажите доступные со стороны каждой роли отношения.

**1. Мониторинг операций в БД**

*Создание таблицы-лога*

Создали таблицу для логирования всех операций в основных таблицах. Эта таблица будет хранить информацию о каждом изменении, включая тип операции (вставка, обновление, удаление), время изменения, пользователя, инициировавшего операцию, и измененные данные:

CREATE TABLE public.main\_log (

log\_item\_id SERIAL PRIMARY KEY,

operation\_type TEXT,

operation\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_operator TEXT,

changed\_data JSONB

);

*Создание функции логирования*

Создали функцию, которая записывает все изменения в таблицу-лог. Она автоматически срабатывает при выполнении операций вставки, удаления или обновления данных в основных таблицах:

CREATE OR REPLACE FUNCTION logging() RETURNS TRIGGER AS $$

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO public.main\_log (operation\_type, operation\_date, user\_operator, changed\_data)

VALUES ('DELETE', now(), current\_user, row\_to\_json(OLD));

ELSIF (TG\_OP = 'UPDATE') THEN

INSERT INTO public.main\_log (operation\_type, operation\_date, user\_operator, changed\_data)

VALUES ('UPDATE', now(), current\_user, row\_to\_json(NEW));

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO public.main\_log (operation\_type, operation\_date, user\_operator, changed\_data)

VALUES ('INSERT', now(), current\_user, row\_to\_json(NEW));

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

*Создание триггеров для таблиц*

Создали триггеры для мониторинга операций в таблицах students, teachers, courses, и других. Эти триггеры вызывают функцию логирования при каждом изменении данных в таблицах, обеспечивая запись всех операций:

CREATE TRIGGER students\_logging\_trigger

AFTER INSERT OR UPDATE OR DELETE ON students

FOR EACH ROW EXECUTE FUNCTION logging();

CREATE TRIGGER teachers\_logging\_trigger

AFTER INSERT OR UPDATE OR DELETE ON teachers

FOR EACH ROW EXECUTE FUNCTION logging();

**2. Шифрование данных**

*Создание таблицы с секретными данными*

Создали таблицу для хранения чувствительной информации, такой как токены доступа. Эта таблица не связана с основными сущностями, чтобы минимизировать риски утечки данных:

CREATE TABLE public.secret\_data (

id SERIAL PRIMARY KEY,

username TEXT,

secret\_token TEXT

);

*Вставка зашифрованных данных*

Для шифрования данных использовали расширение pgcrypto, которое предоставляет средства для симметричного шифрования. В качестве ключа шифрования использовали хэш-функцию от индивидуального пароля, что позволяет защитить данные даже в случае компрометации системы:

CREATE EXTENSION IF NOT EXISTS pgcrypto;

INSERT INTO secret\_data (username, secret\_token)

VALUES (

'example\_user',

pgp\_sym\_encrypt('секретный\_токен', encode(digest('superuser\_password', 'sha256'), 'hex'))

);

*Тестирование доступа к секретным данным*

Для проверки шифрования данных попытались расшифровать данные. Без знания правильного ключа данные остаются недоступными, что подтверждает эффективность шифрования:

SELECT username, pgp\_sym\_decrypt(secret\_token::bytea, encode(digest('superuser\_password', 'sha256'), 'hex')) AS decrypted\_token

FROM secret\_data;

**3. Разграничение доступа в БД**

*Создание ролей и привилегий*

Создали роли для различных типов пользователей: студентов и преподавателей. Для каждой роли были установлены минимальные привилегии, которые обеспечивают доступ только к необходимым данным. Это позволяет снизить риск случайных или злонамеренных изменений данных:

CREATE ROLE student\_group\_role WITH

NOLOGIN

NOSUPERUSER

NOCREATEDB

NOCREATEROLE

INHERIT;

GRANT SELECT ON TABLE public.courses TO student\_group\_role;

CREATE ROLE teacher\_group\_role WITH

NOLOGIN

NOSUPERUSER

NOCREATEDB

NOCREATEROLE

INHERIT;

GRANT SELECT, INSERT, UPDATE ON TABLE public.schedule TO teacher\_group\_role;

*Создание пользователей и присвоение ролей*

Создали пользователей и присвоили им соответствующие роли. Это обеспечивает, что студенты и преподаватели могут выполнять только разрешенные им действия:

CREATE ROLE student\_user WITH

LOGIN

NOSUPERUSER

NOCREATEDB

NOCREATEROLE;

GRANT student\_group\_role TO student\_user;

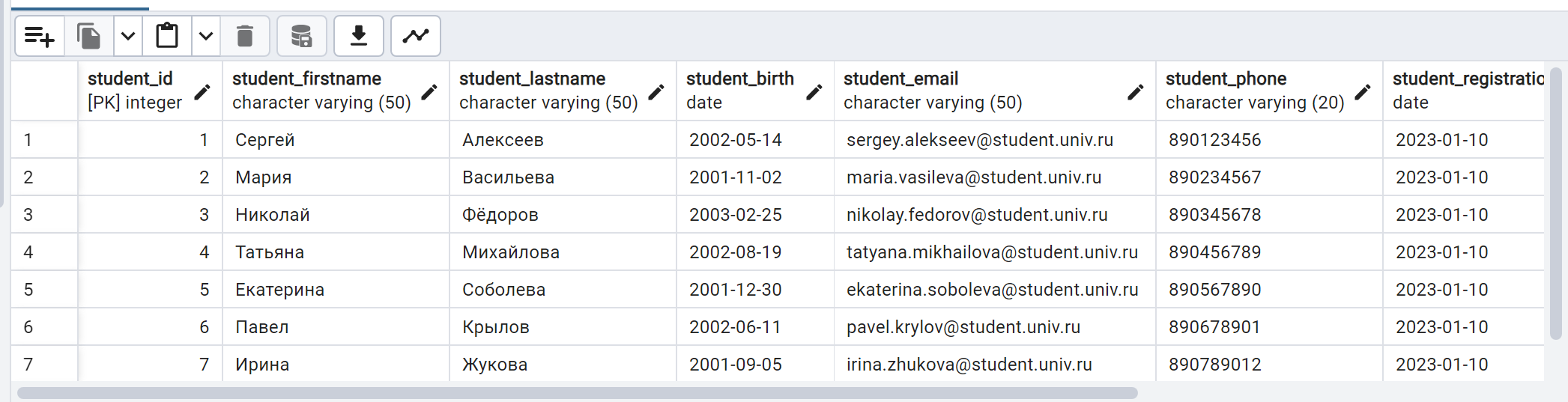
**4. Проверка работы системы**

Для проверки работы системы и получения скриншотов выполниv следующие операции:

1. **Вставка данных в таблицу students**:

INSERT INTO students (student\_firstname, student\_lastname, student\_birth, student\_email, student\_phone, student\_registrationdate)

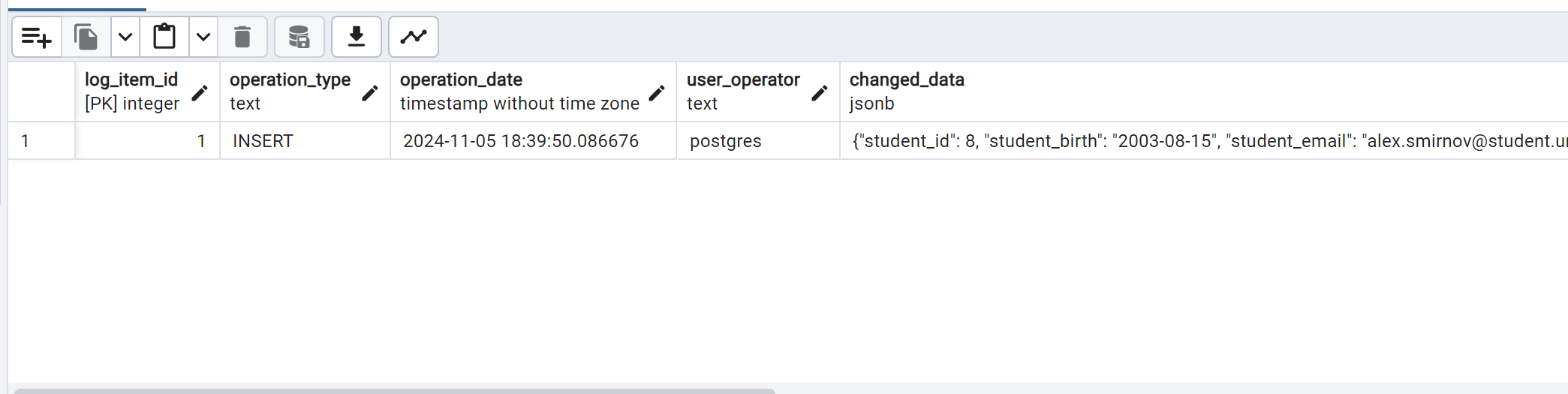
VALUES ('Александр', 'Смирнов', '2003-08-15', 'alex.smirnov@student.univ.ru', '89101112233', '2024-01-15');

****

**Рисунок 1 – Таблица students**

**Проверка таблицы логов** (main\_log):

SELECT \* FROM main\_log;

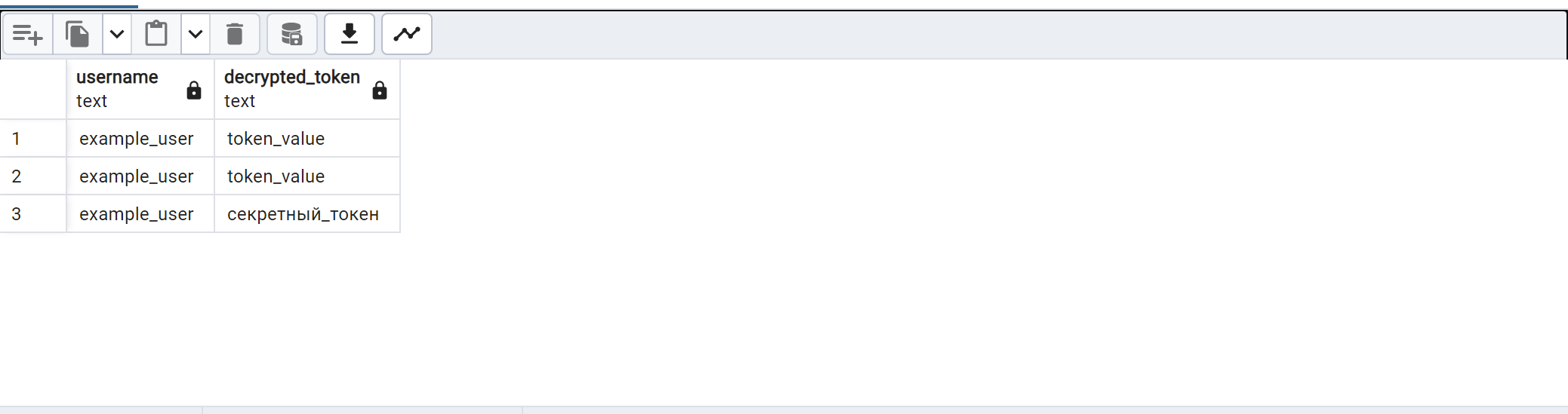
****

**Рисунок 2 – Таблица main\_log**

1. **Расшифровка данных из таблицы secret\_data**:

SELECT username, pgp\_sym\_decrypt(secret\_token::bytea, encode(digest('superuser\_password', 'sha256'), 'hex')) AS decrypted\_token

FROM secret\_data;

****

**Рисунок 3 – таблица secret\_data**

**Заключение:**  
В данной лабораторной работе были выполнены задачи по мониторингу операций в БД, шифрованию чувствительных данных, а также созданию системы разграничения доступа. Было продемонстрировано создание таблицы-лога и триггеров для мониторинга операций, что позволило отслеживать все изменения в базе данных. Шифрование данных с использованием симметричного алгоритма и ключа, зависящего от индивидуального пароля, показало свою эффективность, предотвращая несанкционированный доступ. Система разграничения доступа была успешно реализована путем создания различных ролей и назначения им минимально необходимых привилегий. Проведенное тестирование подтвердило корректность работы всех компонентов системы безопасности и продемонстрировало их высокую эффективность в защите данных от несанкционированного доступа.