МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: Информационная безопасность**

**Дисциплина:**

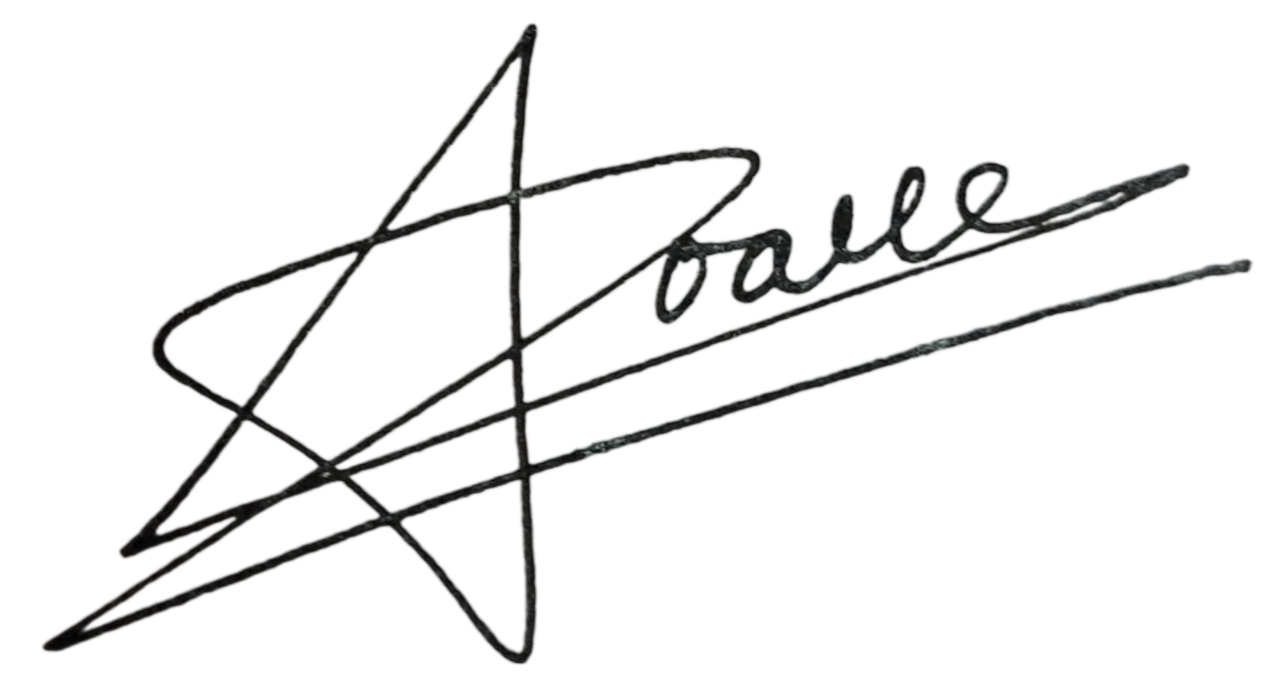
**«Информационная безопасность баз данных»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

***«Защита базы данных»***

**Выполнил студент(ы):**

группа/поток: N3347 / ИББД.Nб3 1.5



\_\_\_\_\_\_\_\_Чу Ван Доан \_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

**Проверил:**

Салихов Максим Русланович \_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

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

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

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

*Дата*

Санкт-Петербург

2024 г.

**СОДЕРЖАНИЕ**

[**Цель работы 3**](#_heading=h.dsn0t69papw)

[**Задание 4**](#_heading=h.b39kds37v2hx)

[**Ход Работы 5**](#_heading=h.fj4jd1a9mogq)

[1. Задачи по мониторингу БД: 5](#_heading=h.91ry3f5v1g9t)

[1.1 Создайте таблицу-лог, отдельную от ваших основных сущностей БД. 5](#_heading=h.dlq2g5k62omj)

[1.2 Создание триггерной функции 5](#_heading=h.5zj933bjtx36)

[1.3 Проверка системы логирования 6](#_heading=h.8wy2gq3cn1f7)

[2. Задачи по шифрованию данных. 7](#_heading=h.h2b7mdl2qq09)

[2.1 Создание таблицы для хранения конфиденциальных данных 7](#_heading=h.nxkx7eqtw5gz)

[2.2 Шифрование данных 7](#_heading=h.56heyb4a035)

[2.3 Создание ключа шифрования из пароля 7](#_heading=h.ytmo9bo5ohf3)

[2.4 Шифрование данных 8](#_heading=h.7oqfv7b47v1h)

[2.5 Доказательство расшифровки данных 8](#_heading=h.lcu5yhgmsr3g)

[3. Задачи по разграничению доступа в БД: 8](#_heading=h.1ksdexcppgrr)

[3.1 Создание роли 8](#_heading=h.7o05qsl7ewr0)

[3.2 Назначение привилегий 9](#_heading=h.h3bzu2tjoa2a)

[3.3 Доступ к таблицам логов 9](#_heading=h.mmyvezjvx5xy)

[3.4 Проведение проверки прав доступа 9](#_heading=h.d13fuq6cser8)

[**Вывод 11**](#_heading=h.m9s4ijudrmu8)

## Цель работы

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

## Задание

1. 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.

1. 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

1. 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.

## Ход Работы

### Задачи по мониторингу БД:

#### 1.1 Создайте таблицу-лог, отдельную от ваших основных сущностей БД.

* Чтобы контролировать базу данных, мы создаем таблицу логов для записи изменений в базе данных. Используем триггеры для автоматического обновления логов каждый раз, когда выполняются операции добавления, изменения или удаления в основных таблицах.

**CREATE TABLE main\_log (**

**log\_id SERIAL PRIMARY KEY,**

**operation\_type TEXT,**

**operation\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**user\_operator TEXT,**

**changed\_data JSON**

**);**

#### 1.2 Создание триггерной функции

* Функция logging() ниже будет записывать информацию в таблицу main\_log.

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

**BEGIN**

**IF (TG\_OP = 'DELETE') THEN**

**INSERT INTO main\_log (operation\_type, user\_operator, changed\_data)**

**VALUES ('D', current\_user, row\_to\_json(OLD));**

**ELSIF (TG\_OP = 'UPDATE') THEN**

**INSERT INTO main\_log (operation\_type, user\_operator, changed\_data)**

**VALUES ('U', current\_user, row\_to\_json(NEW));**

**ELSIF (TG\_OP = 'INSERT') THEN**

**INSERT INTO main\_log (operation\_type, user\_operator, changed\_data)**

**VALUES ('I', current\_user, row\_to\_json(NEW));**

**END IF;**

**RETURN NULL;**

**END;**

**$logging$ LANGUAGE plpgsql;**

* Примените триггер к таблице Product.

**CREATE TRIGGER product\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Bill.

**CREATE TRIGGER bill\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Orders.

**CREATE TRIGGER order\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Customer.

**CREATE TRIGGER customer\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Employee.

**CREATE TRIGGER employee\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Warehouse.

**CREATE TRIGGER warehouse\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

* Примените триггер к таблице Supplier.

**CREATE TRIGGER supplier\_logging**

**AFTER INSERT OR UPDATE OR DELETE ON Product**

**FOR EACH ROW EXECUTE FUNCTION logging();**

#### 1.3 Проверка системы логирования

* Добавление новых данных в таблицу Product:

INSERT INTO product (product\_category\_name, price, warehouse\_id)

VALUES ('Arabica Vip Pro', 200000, 1);

* Обновление данных в таблице Product:

**UPDATE product**

**SET price = 95000**

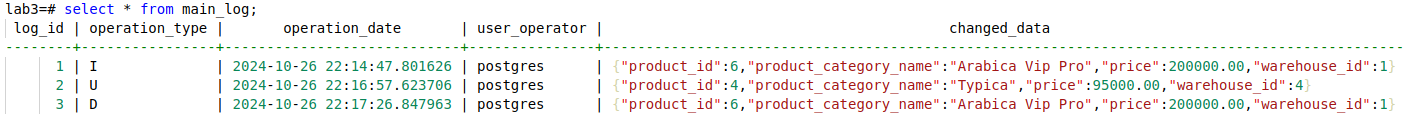
**WHERE product\_id = 4;**

* Удаление данных из таблицы Product:

**DELETE FROM product**

**WHERE product\_id = 6;**

* Проверка main\_log



### Задачи по шифрованию данных.

#### 2.1 Создание таблицы для хранения конфиденциальных данных

* **Сначала нам нужно создать отдельную таблицу для хранения конфиденциальных данных, таких как токены или ключи доступа для каждого типа пользователей. Эта таблица должна быть отделена от основных таблиц базы данных.**

**CREATE TABLE secret\_data (**

**id SERIAL PRIMARY KEY,**

**username TEXT,**

**secret\_token TEXT**

**);**

#### 2.2 **Шифрование данных**

* Мы будем использовать симметричный алгоритм шифрования для шифрования данных в таблице secret\_data. Это гарантирует, что даже если данные будут утечены, их нельзя будет прочитать без ключа шифрования. Подготовка к использованию модуля pgcrypto: PostgreSQL имеет модуль pgcrypto, который поддерживает шифрование и расшифровку данных.

**CREATE EXTENSION IF NOT EXISTS pgcrypto;**

#### 2.3 **Создание ключа шифрования из пароля**

* Khóa mã hóa sẽ được tạo từ mật khẩu của superuser thông qua một hàm băm (ví dụ: SHA-256). Điều này đảm bảo rằng khóa không được lưu trữ trực tiếp trong cơ sở dữ liệu.

**lab3=# select encode(digest('Ch.u992mvd', 'sha256'), 'hex') as encryption\_key;**

**encryption\_key**

**------------------------------------------------------------------**

**ca69b9601669b11f98acf29d694ec0e6d52f581bef9ffbe01bf426a3c2e6418a**

**(1 row)**

#### **2.4 Шифрование данных**

* При вставке данных в таблицу secret\_data мы будем шифровать значение secret\_token с помощью ключа, созданного на предыдущем этапе.

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

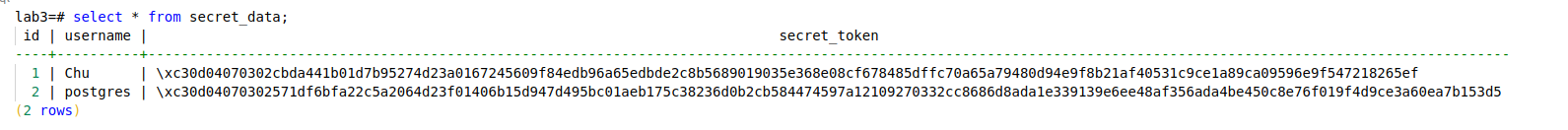
VALUES

('Chu',pgp\_sym\_encrypt('token\_Chu', 'ca69b9601669b11f98acf29d694ec0e6d52f581bef9ffbe01bf426a3c2e6418a')),

('postgres',pgp\_sym\_encrypt('token\_postgres', 'ca69b9601669b11f98acf29d694ec0e6d52f581bef9ffbe01bf426a3c2e6418a'));

#### **2.5 Доказательство расшифровки данных**

* Чтобы доказать, что даже администраторы не могут видеть данные в таблице без знания пароля, мы будем использовать метод расшифровки. Для расшифровки данных нам нужен исходный пароль. Без этого пароля администраторы не смогут расшифровать и прочитать содержимое таблицы secret\_data



* Расшифровка данных:

SELECT username, pgp\_sym\_decrypt(secret\_token::bytea,

'ca69b9601669b11f98acf29d694ec0e6d52f581bef9ffbe01bf426a3c2e6418a')

AS decrypted\_token FROM secret\_data;

username | decrypted\_token

----------+-----------------

Chu | token\_Chu

postgres | token\_postgres

(2 rows)

### Задачи по разграничению доступа в БД:

#### 3.1 Создание роли

* Сначала я создаю две разные роли для различных типов пользователей (не считая суперпользователя): роль sales\_role (роль продавца) и роль warehouse\_role (роль сотрудника склада).

**-- Создание роли для продавца**

**CREATE ROLE sales\_role NOLOGIN;**

**-- Создание роли для сотрудника склада**

**CREATE ROLE warehouse\_role NOLOGIN;**

#### 3.2 Назначение привилегий

* Предоставление прав доступа к схеме:

**GRANT USAGE ON SCHEMA coffee\_shop\_schema TO sales\_role;**

**GRANT USAGE ON SCHEMA coffee\_shop\_schema TO warehouse\_role;**

* **О правах:**
* **Сотрудник продаж (sales\_role): Имеет доступ к просмотру таблиц, связанных с заказами и клиентами.**
* **Сотрудник склада (warehouse\_role): Имеет доступ к просмотру и управлению таблицами продуктов и складов.**

**-- Предоставление прав доступа сотруднику продаж**

**GRANT SELECT ON Orders TO sales\_role;**

**GRANT SELECT ON Customer TO sales\_role;**

**-- Предоставление прав доступа сотруднику склада**

**GRANT SELECT, INSERT, UPDATE ON Product TO warehouse\_role;**

**GRANT SELECT, INSERT ON Warehouse TO warehouse\_role;**

#### 3.3 Доступ к таблицам логов

* Только суперпользователь имеет право просматривать таблицу логов, что помогает защищать конфиденциальную информацию. Вам необходимо убедиться, что никто, кроме суперпользователя, не может получить доступ к этой таблице логов.

**-- Не предоставлять права другим ролям на просмотр таблицы логов**

**REVOKE ALL ON main\_log FROM sales\_role;**

**REVOKE ALL ON main\_log FROM warehouse\_role;**

#### 3.4 Проведение проверки прав доступа

* Вход с ролью sales\_role:

**lab3=# set role sales\_role;**

**SET**

**lab3=> select \* from orders;**

**order\_id | order\_date | total\_amount | employee\_id | customer\_id**

**----------+------------+--------------+-------------+-------------**

**1 | 2024-10-12 | 150000.00 | 2 | 1**

**2 | 2024-10-13 | 599999.00 | 2 | 2**

**3 | 2024-10-13 | 70000.00 | 3 | 3**

**4 | 2024-10-20 | 6699999.00 | 4 | 4**

**(4 rows)**

**lab3=> select \* from product;**

**ERROR: permission denied for table product**

* Вход с ролью warehouse\_role:

**lab3=> SET ROLE warehouse\_role;**

**SET**

**lab3=> SELECT \* FROM Product;**

**product\_id | product\_category\_name | price | warehouse\_id**

**------------+-----------------------+-----------+--------------**

**1 | Arabica | 100000.00 | 1**

**2 | Robusta | 90000.00 | 2**

**3 | Bourbon | 96000.00 | 3**

**4 | Typica | 95000.00 | 4**

**(4 rows)**

* **Проверка доступа к таблице логов:**

**lab3=> SELECT \* FROM Orders;**

**ERROR: permission denied for table orders**

**lab3=> set role sales\_role;**

**SET**

**lab3=> SELECT \* FROM main\_log;**

**ERROR: permission denied for table main\_log**

**lab3=> SET ROLE warehouse\_role;**

**SET**

**lab3=> SELECT \* FROM main\_log;**

**ERROR: permission denied for table main\_log**

**lab3=>**

#### 

## Вывод

В ходе выполнения лабораторной работы по разграничению доступа в базе данных были достигнуты следующие результаты:

1. **Создание ролей**: Были успешно созданы две роли — sales\_role для сотрудников, занимающихся продажами, и warehouse\_role для работников склада. Эти роли были определены в соответствии с потребностями различных классов пользователей информационной системы.
2. **Определение привилегий**: Для каждой роли были установлены четкие привилегии на доступ к таблицам базы данных. Принцип минимальных привилегий был применен, что означает, что каждой роли были предоставлены только необходимые права доступа к определенным таблицам, а доступ к другим таблицам был ограничен.
3. **Ограничения доступа к таблицам логирования**: Таблица логирования main\_log была настроена так, что доступ к ней был предоставлен только суперпользователю, что обеспечило защиту от несанкционированного доступа к данным.
4. **Тестирование и демонстрация**: Были проведены тесты на доступ к данным для каждой роли. Проверки показали, что сотрудники, имеющие роль sales\_role, смогли получить доступ только к таблицам, связанным с заказами и клиентами, в то время как сотрудники со статусом warehouse\_role имели доступ к таблицам продуктов и склада. Попытки доступа к данным из таблицы логирования были успешно заблокированы для всех ролей, кроме суперпользователя.
5. **Выводы о безопасности**: Настройка разграничения доступа позволила создать надежную систему безопасности, предотвращающую несанкционированный доступ к данным. Внедрение ролей и управление привилегиями обеспечивает защиту конфиденциальной информации и соблюдение принципа минимальных привилегий.

Таким образом, лабораторная работа по разграничению доступа в базе данных была успешно выполнена, что продемонстрировало эффективность созданной системы безопасности.