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 №3

«Защита базы данных»

Выполнили:

Бардышев Артём Антонович,

студент группы N3346

_____

(подпись)

Проверил:

Салихов Максим Русланович,

преподаватель, ФБИТ

_____

(отметка о выполнении)

_____

(подпись)

Санкт-Петербург

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – получение навыков организации систем защиты баз данных: логирование операций, разграничение доступа и защита данных методами шифрования.

Введение

Современные системы управления базами данных предоставляют встроенные механизмы для обеспечения информационной безопасности. Среди них можно выделить:

- средства мониторинга (логирование действий пользователей и фиксация изменений);
- разграничение доступа с использованием ролей и привилегий;
- криптографическую защиту чувствительных данных.

В данной работе исследуются базовые механизмы PostgreSQL, которые позволяют реализовать перечисленные функции в учебной базе данных, разработанной ранее (Apple Store).

1 ХОД ВЫПОЛНЕНИЯ

1.1 Мониторинг (логирование операций)

Создана таблица логов и функция-триггер, которая фиксирует изменения во всех таблицах БД.

```
-- Таблица логов
CREATE TABLE main_log (
   log_item_id SERIAL PRIMARY KEY,
   operation_type TEXT,
   operation_date TIMESTAMP,
   user_operator TEXT,
   changed_data JSONB
);

-- Триггерная функция
CREATE OR REPLACE FUNCTION logging() RETURNS TRIGGER AS $$
BEGIN
   IF (TG_OP = 'DELETE') THEN
      INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
      VALUES ('DELETE', now(), current_user, row_to_json(OLD));
   ELSIF (TG_OP = 'UPDATE') THEN
      INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
      VALUES ('UPDATE', now(), current_user, row_to_json(NEW));
   ELSIF (TG_OP = 'INSERT') THEN
      INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
      VALUES ('INSERT', now(), current_user, row_to_json(NEW));
   END IF;
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

-- Пример для таблицы Customer

CREATE TRIGGER trg_customer_log

AFTER INSERT OR UPDATE OR DELETE ON Customer

FOR EACH ROW EXECUTE FUNCTION logging();

```
apple_store=# -- Таблица логов
apple_store=# CREATE TABLE main_log (
apple_store(#     log_item_id SERIAL PRIMARY KEY,
apple_store(#     operation_type TEXT,
apple_store(#     operation_date TIMESTAMP,
apple_store(#     user_operator TEXT,
apple_store(#     changed_data JSONB
apple_store(# );
CREATE TABLE
apple_store=#
apple_store=# -- Триггерная функция
apple_store=# CREATE OR REPLACE FUNCTION logging() RETURNS TRIGGER AS $$
apple_store$# BEGIN
apple_store$#     IF (TG_OP = 'DELETE') THEN
apple_store$#         INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
apple_store$#         VALUES ('DELETE', now(), current_user, row_to_json(OLD));
apple_store$#     ELSIF (TG_OP = 'UPDATE') THEN
apple_store$#         INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
apple_store$#         VALUES ('UPDATE', now(), current_user, row_to_json(NEW));
apple_store$#     ELSIF (TG_OP = 'INSERT') THEN
apple_store$#         INSERT INTO main_log (operation_type, operation_date, user_operator, changed_data)
apple_store$#         VALUES ('INSERT', now(), current_user, row_to_json(NEW));
apple_store$#     END IF;
apple_store$#     RETURN NULL;
apple_store$# END;
apple_store$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
apple_store=#
apple_store=# -- Пример для таблицы Customer
apple_store=# CREATE TRIGGER trg_customer_log
apple_store-# AFTER INSERT OR UPDATE OR DELETE ON Customer
apple_store-# FOR EACH ROW EXECUTE FUNCTION logging();
CREATE TRIGGER
```

1.2 Шифрование секретных данных

Подключаем расширение:

CREATE EXTENSION IF NOT EXISTS pgcrypto;

```
apple_store=# CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE EXTENSION
```

Создаём таблицу:

CREATE TABLE secret_data (

id SERIAL PRIMARY KEY,

username TEXT,

secret_token BYTEA

);

```
apple_store=# CREATE TABLE secret_data (
apple_store(#     id SERIAL PRIMARY KEY,
apple_store(#     username TEXT,
apple_store(#     secret_token BYTEA
apple_store(# );
CREATE TABLE
```

Генерация ключа (SHA-256 от пароля !stroNgpsw31234):

-- пример вставки зашифрованных данных

INSERT INTO secret_data (username, secret_token) VALUES

(

'admin_user',

pgp_sym_encrypt('token_ABC123',

'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')

),

(

'staff_user',

pgp_sym_encrypt('token_XYZ789',

'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')

);

```
apple_store=# -- пример вставки зашифрованных данных
apple_store=# INSERT INTO secret_data (username, secret_token) VALUES
apple_store-# (
apple_store(#     'admin_user',
apple_store(#     pgp_sym_encrypt('token_ABC123', '9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')
apple_store(# ),
apple_store-# (
apple_store(#     'staff_user',
apple_store(#     pgp_sym_encrypt('token_XYZ789', '9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')
apple_store(# );
INSERT 0 2
```

Расшифровка доступна только при знании пароля:

SELECT username, pgp_sym_decrypt(secret_token,

'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')

AS token FROM secret_data;

```
apple_store=# SELECT username, pgp_sym_decrypt(secret_token,
apple_store(# '9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')
apple_store-# AS token FROM secret_data;
  username  |    token
------------+-------------
 admin_user | token_ABC123
 staff_user | token_XYZ789
(2 строки)
```

1.3 Разграничение доступа

Создаём три групповые роли: **student_group_role, staff_group_role, admin_group_role**.

-- Студенты: могут только читать часть представлений

CREATE ROLE student_group_role NOLOGIN;

GRANT SELECT ON compact_shedule, consultations TO student_group_role;

-- Персонал: доступ к другим представлениям

CREATE ROLE staff_group_role NOLOGIN;

GRANT SELECT ON speciality_and_contacts, speciality_and_teachers TO staff_group_role;

-- Админ: полный доступ

CREATE ROLE admin_group_role NOLOGIN;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO admin_group_role;

```
apple_store=# -- Студенты: могут только читать часть представлений
apple_store=# CREATE ROLE student_group_role NOLOGIN;
CREATE ROLE
apple_store=# GRANT SELECT ON compact_shedule, consultations TO student_group_role;
ОШИБКА:  отношение "compact_shedule" не существует
apple_store=#
apple_store=# -- Персонал: доступ к другим представлениям
apple_store=# CREATE ROLE staff_group_role NOLOGIN;
CREATE ROLE
apple_store=# GRANT SELECT ON speciality_and_contacts, speciality_and_teachers TO staff_group_role;
ОШИБКА:  отношение "speciality_and_contacts" не существует
apple_store=#
apple_store=# -- Админ: полный доступ
apple_store=# CREATE ROLE admin_group_role NOLOGIN;
CREATE ROLE
apple_store=# GRANT SELECT ON ALL TABLES IN SCHEMA public TO admin_group_role;
GRANT
```

Создаём индивидуальных пользователей:

CREATE ROLE ivan_student LOGIN PASSWORD 'studpass';

GRANT student_group_role TO ivan_student;

CREATE ROLE elena_staff LOGIN PASSWORD 'staffpass';

GRANT staff_group_role TO elena_staff;

CREATE ROLE petya_admin LOGIN PASSWORD 'adminpass';

GRANT admin_group_role TO petya_admin;

```
apple_store=# GRANT student_group_role TO ivan_student;
GRANT ROLE
apple_store=#
apple_store=# CREATE ROLE elena_staff LOGIN PASSWORD 'staffpass';
CREATE ROLE
apple_store=# GRANT staff_group_role TO elena_staff;
GRANT ROLE
apple_store=#
apple_store=# CREATE ROLE petya_admin LOGIN PASSWORD 'adminpass';
CREATE ROLE
apple_store=# GRANT admin_group_role TO petya_admin;
GRANT ROLE
```

Проверка:

SET ROLE ivan_student;

SELECT * FROM compact_shedule; -- работает

SELECT * FROM Customer; -- ошибка доступа

```
apple_store=> SELECT * FROM compact_shedule; -- работает
 orderid | orderdate  |   status   | totalsum  | itemscount | lastpaymentstatus
---------+------------+------------+-----------+------------+-------------------
       4 | 2024-05-13 | completed  | 200000.00 |          1 | success
       6 | 2024-05-15 | processing | 115000.00 |          1 | pending
       2 | 2024-05-11 | processing | 140000.00 |          1 | pending
       3 | 2024-05-12 | cancelled  |  60000.00 |          1 | refunded
       5 | 2024-05-14 | completed  | 100000.00 |          1 | success
       1 | 2024-05-10 | completed  | 210000.00 |          2 | success
(6 строк)


apple_store=> SELECT * FROM Customer; -- ошибка доступа
ОШИБКА:  нет доступа к таблице customer
```

При создании некоторых ролей и зависимостей возникли некоторые трудности, внимательно разобравшись и войдя в профиль суперюзера root, было пересоздано все что не удалось и скриншот выше – итоговый правильный результат.

ЗАКЛЮЧЕНИЕ

В ходе работы разработана система защиты БД средствами PostgreSQL.

- Реализовано логирование операций с данными с помощью триггеров.
- Создана таблица секретных данных с симметричным шифрованием на основе pgcrypto.
- Настроено разграничение доступа на основе ролей пользователей, реализован принцип минимальных привилегий.

Таким образом, выполнены базовые задачи по обеспечению безопасности базы данных: контроль действий, защита чувствительных данных и управление правами доступа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Новиков Б. А., Горшкова Е. А., Графеева Н. Г. **Основы технологий баз данных.** – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с.
2. Хомоненко А. Д. (ред.). **Базы данных.** – 6-е изд., доп. – СПб.: КОРОНА-Век, 2009. – 736 с.
3. Документация PostgreSQL: https://www.postgresql.org/docs/
4. Модуль `pgcrypto`: https://www.postgresql.org/docs/current/pgcrypto.html