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

Факультет безопасности информационных технологий

Направление подготовки: 11.03.03
Образовательная программа: Безопасность информационных технологий

Дисциплина:
«Информационная безопасность баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
« Защита базы данных »

Выполнил:
Рядовой Т.С., студент группы N3352, поток ИББД.N63 1.5

(подпись)

Проверил:
Салихов Максим Русланович

(отметка о выполнении)

(подпись)

(дата)

Санкт-Петербург
2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: получение навыков создания примитивных систем мониторинга, разграничения доступа и шифрования средствами СУБД.

1 ХОД РАБОТЫ

1.1 Задача №1 – мониторинг БД

Задачи:

- Создать таблицу-лог для записи изменений в БД;
- Создать триггеры для основных таблиц, которые будут фиксировать изменения (вставка, обновление, удаление) и записывать их в таблицу-лог;
- Продемонстрировать работу системы логирования.

1.1.1 Создание лог-таблицы

Листинг 1 – Создание таблицы

```
CREATE TABLE public.main_log (
    log_item_id SERIAL PRIMARY KEY,
    table_name VARCHAR(50) NOT NULL,
    operation_type VARCHAR(30) NOT NULL,
    operation_date TIMESTAMP,
    user_operator VARCHAR(30) NOT NULL,
    changed_data JSONB
);
```

1.1.2 Создание функции и триггеров

Листинг 2 – Создание функции

```
CREATE OR REPLACE FUNCTION logging() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO public.main_log (table_name, operation_type,
operation_date, user_operator, changed_data)
    VALUES (
        TG_TABLE_NAME,
        TG_OP,
        NOW(),
        current_user,
        row_to_json(CASE WHEN TG_OP = 'DELETE' THEN OLD ELSE NEW END)
    );
    RETURN CASE WHEN TG_OP = 'DELETE' THEN OLD ELSE NEW END;
END;
$$ LANGUAGE plpgsql;
```

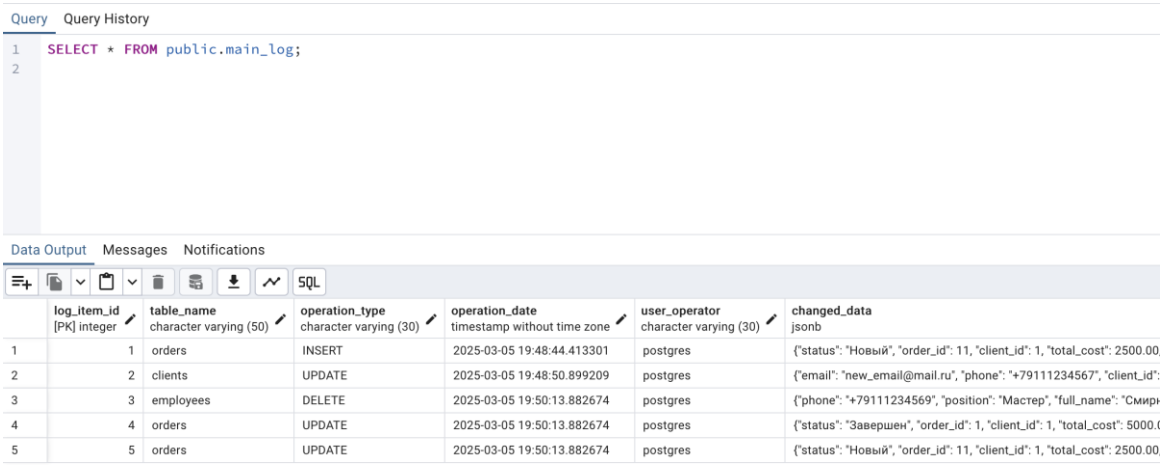
Листинг 3 – Создание триггеров

```
-- Триггер для таблицы "Заказы"
CREATE TRIGGER logging_orders
AFTER INSERT OR UPDATE OR DELETE ON public.orders
FOR EACH ROW EXECUTE FUNCTION logging();

-- Триггер для таблицы "Клиенты"
CREATE TRIGGER logging_clients
AFTER INSERT OR UPDATE OR DELETE ON public.clients
FOR EACH ROW EXECUTE FUNCTION logging();

-- Триггер для таблицы "Сотрудники"
CREATE TRIGGER logging_employees
```

```
AFTER INSERT OR UPDATE OR DELETE ON public.employees
FOR EACH ROW EXECUTE FUNCTION logging();

-- Вставка данных в таблицу "Заказы"
INSERT INTO public.orders (creation_date, completion_date, status,
total_cost, client_id, employee_id)
VALUES ('2024-02-02', NULL, 'Новый', 2500.00, 1, 1);

-- Обновление данных в таблице "Клиенты"
UPDATE public.clients SET email = 'new_email@mail.ru' WHERE client_id =
1;

-- Удаление данных из таблицы "Сотрудники"
DELETE FROM public.employees WHERE employee_id = 1;

-- Проверка таблицы-лога
SELECT * FROM public.main_log;
```

1.1.3 Пример работы

Видим, что также обновились записи в таблице orders, где был удаленный employee_id.



	log_item_id [PK] integer	table_name character varying (50)	operation_type character varying (30)	operation_date timestamp without time zone	user_operator character varying (30)	changed_data jsonb
1	1	orders	INSERT	2025-03-05 19:48:44.413301	postgres	{"status": "Новый", "order_id": 11, "client_id": 1, "total_cost": 2500.00
2	2	clients	UPDATE	2025-03-05 19:48:50.899209	postgres	{"email": "new_email@mail.ru", "phone": "+79111234567", "client_id":
3	3	employees	DELETE	2025-03-05 19:50:13.882674	postgres	{"phone": "+79111234569", "position": "Мастер", "full_name": "Смир
4	4	orders	UPDATE	2025-03-05 19:50:13.882674	postgres	{"status": "Завершен", "order_id": 1, "client_id": 1, "total_cost": 5000.(
5	5	orders	UPDATE	2025-03-05 19:50:13.882674	postgres	{"status": "Новый", "order_id": 11, "client_id": 1, "total_cost": 2500.00

Рисунок 1 – Пример заполнения логов после срабатывания триггеров

	order_id [PK] integer	creation_date date	completion_date date	status character varying (50)	total_cost numeric (10,2)	client_id integer	employee_id integer
1	2	2024-01-05	[null]	В процессе	3000.00	2	2
2	3	2024-01-10	[null]	Новый	2000.00	3	3
3	4	2024-02-01	[null]	Новый	4000.00	4	4
4	5	2024-02-05	2024-02-15	Завершен	3500.00	5	5
5	6	2024-02-10	[null]	В процессе	2500.00	6	6
6	7	2024-02-15	[null]	Новый	1500.00	7	7
7	8	2024-02-20	[null]	Новый	1800.00	8	8
8	9	2024-02-25	2024-03-01	Завершен	5000.00	9	9
9	10	2024-03-01	[null]	В процессе	2200.00	10	10
10	1	2024-01-01	2024-01-10	Завершен	5000.00	1	[null]
11	11	2024-02-02	[null]	Новый	2500.00	1	[null]

Рисунок 2 – Обновление таблицы orders после удаление employee_id

1.2 Задача №2 – шифрование данных

Задачи по шифрованию данных:

- Создать таблицу с секретными данными;
- Зашифровать данные в таблице с использованием симметричного алгоритма шифрования (например, AES-256);
- Продемонстрировать, что без знания ключа шифрования данные недоступны.

1.2.1 Создание таблицы и шифрование данных

Листинг 4 – Создание таблицы и шифрование

```
CREATE TABLE public.secret_data (
    id SERIAL PRIMARY KEY,
    username VARCHAR(30) NOT NULL,
    secret_token BYTEA NOT NULL
);

-- Установка расширения pgcrypto
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Вставка зашифрованных данных
INSERT INTO public.secret_data (username, secret_token)
VALUES (
    'operator_1', pgp_sym_encrypt('token_',
'9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08')
);
```

1.2.2 Пример работы

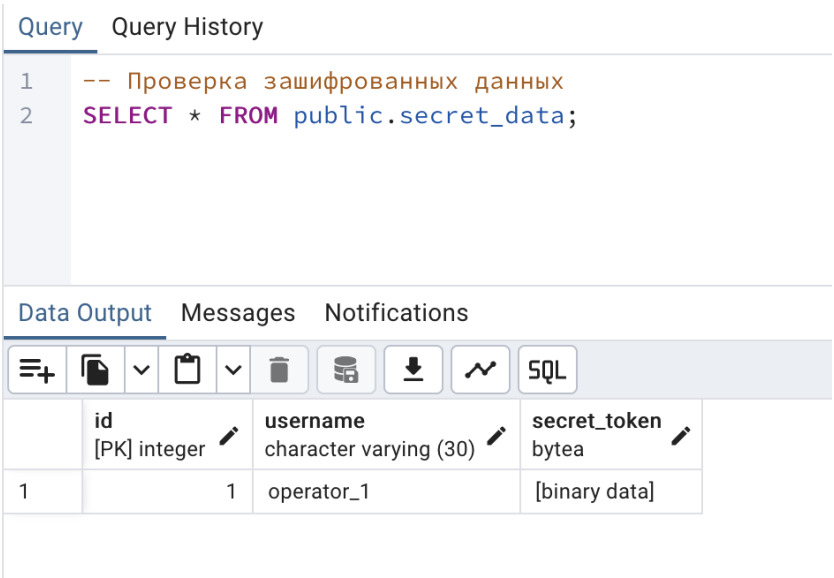


Рисунок 3 – Проверка данных

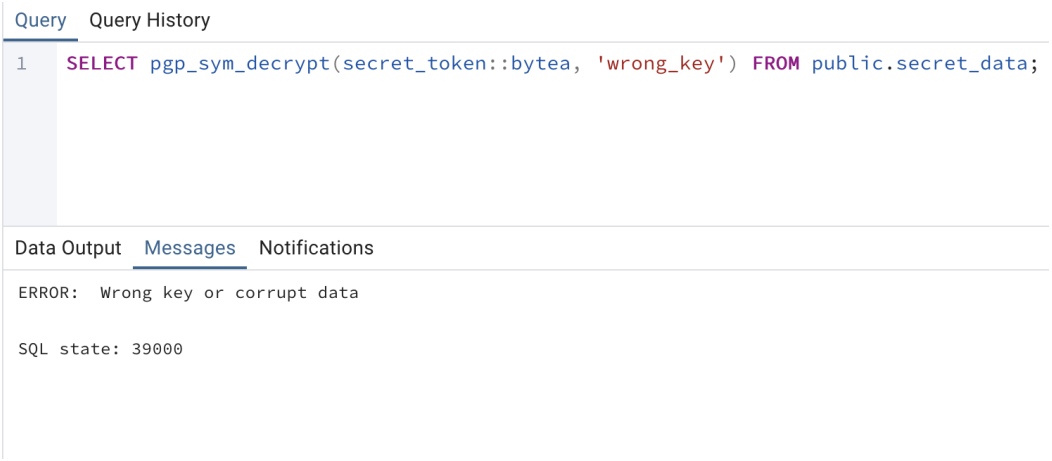


Рисунок 4 – Попытка расшифровки данных без ключа

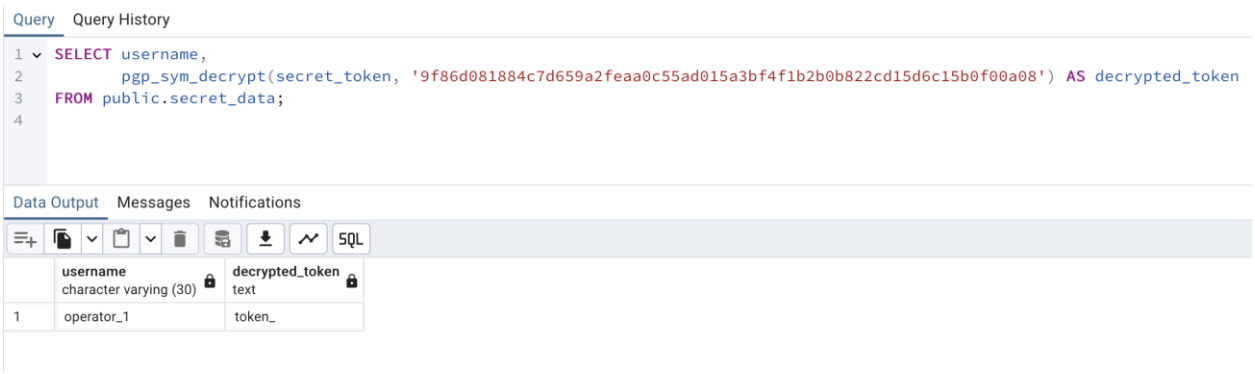


Рисунок 5 – Успешная попытка расшифровки

1.3 Задача №3 – разграничение доступа к БД

Задачи по разграничению доступа в БД:

- Создать роли для различных классов пользователей (например, операторы, мастера);
- Настроить привилегии для каждой роли в соответствии с принципом минимальных привилегий;
- Продемонстрировать работу системы разграничения доступа.

1.3.1 Создание ролей

Листинг 5 – Создание ролей

```
-- Роль для операторов
CREATE ROLE operator_role WITH
    NOLOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    INHERIT;
```

```
-- Роль для мастеров
CREATE ROLE master_role WITH
    NOLOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION
    INHERIT;
```

1.3.2 Настройка привилегий

Листинг 6 – Настройка привилегий

```
-- Привилегии для операторов
GRANT SELECT, INSERT, UPDATE ON public.orders TO operator_role;
GRANT SELECT ON public.clients TO operator_role;

-- Привилегии для мастеров
GRANT SELECT, INSERT, UPDATE ON public.employees TO master_role;
GRANT SELECT ON public.materials TO master_role;

-- Пользователь для оператора
CREATE ROLE operator_user WITH LOGIN PASSWORD 'operator_pass';
GRANT operator_role TO operator_user;

-- Пользователь для мастера
CREATE ROLE master_user WITH LOGIN PASSWORD 'master_pass';
GRANT master_role TO master_user;
```

1.3.3 Пример работы

Query | Query History

```
1  SET ROLE operator_user;
2  SELECT * FROM public.orders;
```

Data Output | Messages | Notifications

	order_id [PK] integer	creation_date date	completion_date date	status character varying (50)	total_cost numeric (10,2)	client_id integer	employee_id integer
1	2	2024-01-05	[null]	В процессе	3000.00	2	2
2	3	2024-01-10	[null]	Новый	2000.00	3	3
3	4	2024-02-01	[null]	Новый	4000.00	4	4
4	5	2024-02-05	2024-02-15	Завершен	3500.00	5	5
5	6	2024-02-10	[null]	В процессе	2500.00	6	6
6	7	2024-02-15	[null]	Новый	1500.00	7	7
7	8	2024-02-20	[null]	Новый	1800.00	8	8
8	9	2024-02-25	2024-03-01	Завершен	5000.00	9	9
9	10	2024-03-01	[null]	В процессе	2200.00	10	10
10	1	2024-01-01	2024-01-10	Завершен	5000.00	1	[null]
11	11	2024-02-02	[null]	Новый	2500.00	1	[null]

Рисунок 6 – Успешная попытка доступа

Рисунок 7 – Неуспешная попытка доступа

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были реализованы системы мониторинга, шифрования данных и разграничения доступа в базе данных. Эти механизмы обеспечивают безопасность данных и контроль доступа к информации в рамках СУБД.

СПИСОК ИСТОЧНИКОВ

1. Документация PostgreSQL – Создание ролей [Электронный ресурс]. – URL: https://postgrespro.ru/docs/postgresql/9.6/sql-createrole (Дата обращения: 25.01.2025).