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 / Information security"

Дисциплина:

«*Информационная безопасность баз данных*»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«Разграничение доступа»

Выполнил студент:

N3246 / ИББД N23 1.3

Суханкулиев Мухаммет /_____

ФИО *Подпись*

Проверила:

Карманова Наталия Андреевна /_____

ФИО *Подпись*

_____

Отметка о выполнении (один из вариантов: отлично, хорошо, удовлетворительно, зачтено)

_____

Дата

Санкт-Петербург

2025 г.

ВВЕДЕНИЕ

Цель работы – освоить методы разграничения прав доступа и аудита действий пользователей в PostgreSQL.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Подготовить таблицы для выполнения перечисленных ниже задач. Достаточно 2–3 таблиц для п. 1–5 ниже.
2. Выдать права 3 пользователям. Пользователь User1 должен иметь полный доступ к таблице. User2 должен иметь право на вставку, select-запросы и обновление значений в таблицах. User3 должен иметь право на удаление строк из таблиц, а также возможность делегировать свои права любому пользователю.
3. Предоставить право на удаление от пользователя User3 пользователю User4 и проверить все выданные права.
4. Отменить все предоставленные выше права.
5. Создать подсхему авторизации для User1 и User2 с различным набором таблиц (служебное слово AUTHORIZATION у команды CREATE SCHEMA).
6. Создать представление как объединенный набор столбцов из разных таблиц. С помощью команды grant ограничить доступ к представлению.
7. Настроить безопасность на уровне строк (RLS), политика должна быть создана на основе текущего пользователя, и протестировать ее.
8. Создать триггер для регистрации вставки, обновления и удаления содержимого в определенных таблицах.

1 РАЗГРАНИЧЕНИЕ ДОСТУПА

База данных ibbd создана локально. Манипуляции будут производиться с помощью консоли psql и программы pgAdmin 4.

1.1 Подготовка таблиц

Создаём 3 простые таблицы для дальнейшей работы с правами.

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name TEXT,
    position TEXT
);

CREATE TABLE departments (
    id SERIAL PRIMARY KEY,
    name TEXT
);

CREATE TABLE salaries (
    id SERIAL PRIMARY KEY,
    employee_id INT REFERENCES employees(id),
    amount NUMERIC
);

CREATE TABLE Query returned successfully in 106 msec.
```

1.2 Разграничение прав доступа

Создаём пользователей и выдаём им права.

```
CREATE USER User1 WITH PASSWORD 'password1';
CREATE USER User2 WITH PASSWORD 'password2';
CREATE USER User3 WITH PASSWORD 'password3';

CREATE ROLE Query returned successfully in 139 msec.
```

```
GRANT ALL PRIVILEGES ON employees, departments, salaries TO User1;
GRANT SELECT, INSERT, UPDATE ON employees, departments, salaries TO User2;
GRANT DELETE ON employees, departments, salaries TO User3 WITH GRANT OPTION;

GRANT Query returned successfully in 61 msec.
```

WITH GRANT OPTION позволяет User3 передавать свои права другим.

1.3 Предоставление права доступа от одного пользователя другому

Создаём User4 и выдаём права через делегирование.

```
CREATE USER User4 WITH PASSWORD 'password4';

CREATE ROLE Query returned successfully in 63 msec.
```

```
ibbd=# SET ROLE User3;
SET
ibbd=> GRANT DELETE ON employees, departments TO User4;
GRANT
ibbd=> RESET ROLE;
RESET
```

```
ibbd=# \dp employees
                                    Права доступа
 Схема  |    Имя    |   Тип   |       Права доступа        | Права для столбцов |
Политики
--------+-----------+---------+----------------------------+--------------------+----
 public | employees | таблица | postgres=arwdDxtm/postgres+|                    |
        |           |         | user1=arwdDxtm/postgres   +|                    |
        |           |         | user2=arw/postgres        +|                    |
        |           |         | user3=d*/postgres         +|                    |
        |           |         | user4=d/user3              |                    |
(1 строка)


ibbd=# \dp salaries
                                    Права доступа
 Схема  |   Имя    |   Тип   |       Права доступа        | Права для столбцов |
Политики
--------+----------+---------+----------------------------+--------------------+----
 public | salaries | таблица | postgres=arwdDxtm/postgres+|                    |
        |          |         | user1=arwdDxtm/postgres   +|                    |
        |          |         | user2=arw/postgres        +|                    |
        |          |         | user3=d*/postgres          |                    |
(1 строка)
```

1.4 Отмена всех предоставленных прав

Отзываем все права.

```
REVOKE ALL PRIVILEGES ON employees, departments, salaries FROM User1, User2,
User3, User4 CASCADE;

REVOKE Query returned successfully in 69 msec.
```

CASCADE снимает права, включая те, что были переданы другим.

1.5 Подсхемы авторизаций

Создаём разные схемы для User1 и User2.

```
CREATE SCHEMA schema_user1 AUTHORIZATION User1;
CREATE SCHEMA schema_user2 AUTHORIZATION User2;

CREATE SCHEMA Query returned successfully in 67 msec.
```

Пример с таблицами в этих схемах:

```
CREATE TABLE schema_user1.personal_data (
    id SERIAL PRIMARY KEY,
    name TEXT
);

CREATE TABLE schema_user2.task_log (
    id SERIAL PRIMARY KEY,
    task TEXT
);

CREATE TABLE Query returned successfully in 75 msec.
```

```
ibbd=# \dn
           Список схем
     Имя      |     Владелец
--------------+------------------
```

```
 public       | pg_database_owner
 schema_user1 | user1
 schema_user2 | user2
(3 строки)
```

Теперь каждый пользователь будет работать в своей подсхеме.

1.6 Представление

Создаём VIEW, объединяющее данные из нескольких таблиц.

```
CREATE VIEW public.view_employee_salary AS
SELECT e.name, e.position, s.amount
FROM employees e
JOIN salaries s ON e.id = s.employee_id;

CREATE VIEW Query returned successfully in 62 msec.
```

```
REVOKE ALL ON view_employee_salary FROM PUBLIC;
GRANT SELECT ON view_employee_salary TO User2;

GRANT Query returned successfully in 69 msec.
```

Теперь только User2 может читать данные из представления.

```
ibbd=# SET ROLE User1;
SET
ibbd=> select * from view_employee_salary;
ОШИБКА:  нет доступа к представлению view_employee_salary
ibbd=> SET ROLE User2;
SET
ibbd=> select * from view_employee_salary;
name | position | amount
------+----------+--------
(0 строк)
```

1.7 RLS

Включаем контроль доступа на уровне строк, чтобы дать каждому пользователю доступ только к своим данным.

```
ALTER TABLE employees ENABLE ROW LEVEL SECURITY;

ALTER TABLE Query returned successfully in 76 msec.
```

```
ALTER TABLE employees ADD COLUMN owner TEXT;

ALTER TABLE Query returned successfully in 65 msec.
```

```
INSERT INTO employees (name, position, owner) VALUES
('Alice', 'Manager', 'user1'),
('Bob', 'Engineer', 'user2'),
('Carol', 'Analyst', 'user1');

INSERT 0 3 Query returned successfully in 76 msec.
```

```
CREATE POLICY employee_rls_policy
ON employees
FOR SELECT
USING (owner = current_user);
```

```
CREATE POLICY Query returned successfully in 67 msec.
```

```
GRANT SELECT ON employees TO user1, user2;

GRANT Query returned successfully in 68 msec.
```

Проверка:

```
ibbd=# set role User1;
SET
ibbd=> select * from employees;
id | name  | position | owner
----+-------+----------+-------
  1 | Alice | Manager  | user1
  3 | Carol | Analyst  | user1
(2 строки)


ibbd=> set role User2;
SET
ibbd=> select * from employees;
id | name | position | owner
----+------+----------+-------
  2 | Bob  | Engineer | user2
(1 строка)
```

1.8 Триггер

Таблица аудита:

```
CREATE TABLE audit_log (
    id SERIAL PRIMARY KEY,
    username TEXT,
    action TEXT,
    table_name TEXT,
    row_data JSONB,
    action_time TIMESTAMP DEFAULT current_timestamp
);

CREATE TABLE Query returned successfully in 52 msec.
```

Функция триггера:

```
CREATE OR REPLACE FUNCTION log_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'INSERT') THEN
        INSERT INTO audit_log (username, action, table_name, row_data)
        VALUES (current_user, 'INSERT', TG_TABLE_NAME, to_jsonb(NEW));
        RETURN NEW;

    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO audit_log (username, action, table_name, row_data)
        VALUES (current_user, 'UPDATE', TG_TABLE_NAME, to_jsonb(NEW));
        RETURN NEW;

    ELSIF (TG_OP = 'DELETE') THEN
        INSERT INTO audit_log (username, action, table_name, row_data)
        VALUES (current_user, 'DELETE', TG_TABLE_NAME, to_jsonb(OLD));
        RETURN OLD;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE FUNCTION Query returned successfully in 123 msec.
```

Назначение триггера для таблицы:

```
CREATE TRIGGER trg_audit_employees
AFTER INSERT OR UPDATE OR DELETE ON employees
FOR EACH ROW
EXECUTE FUNCTION log_changes();

CREATE TRIGGER Query returned successfully in 61 msec.
```

Проверка:

```
INSERT INTO employees (name, position, owner) VALUES ('TestUser', 'Tester',
'user1');
INSERT 0 1 Query returned successfully in 62 msec.

ibbd=# SELECT * FROM audit_log;
id | username | action | table_name | row_data | action_time
---+----------+--------+------------+----------+-------------
  1 | postgres | INSERT | employees  | {"id": 4, "name": "TestUser", "owner":
"user1", "position": "Tester"} | 2025-04-29 11:24:09.47262
(1 строка)
```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были изучены и практически реализованы механизмы разграничения прав доступа в PostgreSQL, включая права на уровне объектов, делегирование полномочий, безопасность на уровне строк (RLS), работу с подсхемами и представлениями. Работа также включала создание триггера аудита для регистрации действий пользователей.

Это позволило на практике освоить основные средства обеспечения безопасности данных и контроля активности пользователей в СУБД PostgreSQL.