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 №4

«Контроль доступа и системы аудита»

Выполнили:

Чу Ван Доан, студент группы N3247

(подпись)

Проверил:

Волков А.Г.

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

Содержание

1 Контроль доступа и системы аудита

1.1 Цель работы

Получение навыков контроля доступа и системы аудита.

1.2 Задание

1.2.1 Подготовьте таблицы для выполнения перечисленных ниже задач.

1.2.2 Выдайте 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.

1.2.3 Предоставьте право на удаление от пользователя User3 пользователю User4 и проверьте все выданные права.

1.2.4 Отмените все предоставленные выше права.

1.2.5 Создайте подсхему для User1 и User2 с различным набором таблиц. Покажите возможен ли доступ пользователей из одной подсхемы к другой.

1.2.6 Создайте представление как объединенный набор столбцов из разных таблиц. Ограничьте доступ к созданному представлению для User1.

1.2.7 Настройте безопасность на уровне строк, политика должна быть создана на основе текущего пользователя, и протестируйте ее.

1.2.8 Создайте триггер для регистрации вставки, обновления и удаления содержимого в определенных таблицах.

1.3 Ход работы

1.3.1 Задание 1

Создадим таблицу foreign_students:

```
n3247_22_lab4=# create table foreign_students (id bigserial primary key, name varchar(30), age int, country varchar(30));
CREATE TABLE
```

Рисунок 1 – Создание таблицы foreign_students

Вставим записи в таблицу foreign_students:

```
n3247_22_lab4=# insert into foreign_students (name, age, country) values ('Doan', 22, 'VietNam'), ('DamThuong', 20, 'VietNam'), ('Rikaono', 22, 'Japan'), ('Aibeili', 22, 'China');
INSERT 0 4
n3247_22_lab4=# select * from foreign_students;
 id |   name    | age | country
----+-----------+-----+---------
  1 | Doan      |  22 | VietNam
  2 | DamThuong |  20 | VietNam
  3 | Rikaono   |  22 | Japan
  4 | Aibeili   |  22 | China
(4 rows)
```

Рисунок 2 – Вставка записей в таблицу foreign_students

1.3.2 Задание 2

Создадим позьзователей user1, user2, user3:

```
n3247_22_lab4=# create user user1 with password 'pass1';
CREATE ROLE
n3247_22_lab4=# create user user2 with password 'pass2';
CREATE ROLE
n3247_22_lab4=# create user user3 with password 'pass3';
CREATE ROLE
```

Рисунок 3 – Создание пользователей

Дадим права пользователям:

```
n3247_22_lab4=# grant all on foreign_students to user1;
GRANT
n3247_22_lab4=# grant insert, select, update on foreign_students to user2;
GRANT
n3247_22_lab4=# grant delete on foreign_students to user3 with grant option;
GRANT
n3247_22_lab4=# \dp foreign_students;
                                 Access privileges
 Schema |       Name       | Type  |     Access privileges     | Column privileges | Policies
--------+------------------+-------+---------------------------+-------------------+----------
 public | foreign_students | table | postgres=arwdDxt/postgres+|                   |
        |                  |       | user1=arwdDxt/postgres   +|                   |
        |                  |       | user2=arw/postgres       +|                   |
        |                  |       | user3=d*/postgres         |                   |
(1 row)
```

Рисунок 4 – Давание прав пользователям

1.3.3 Задание 3

Создадим позьзователя user4:

```
n3247_22_lab4=# create user user4 with password 'pass4';
CREATE ROLE
```

Рисунок 5 – Создание позователя user4

Предоставим право на удаление от пользователя user3 пользователю user4:

```
n3247_22_lab4=# set role user3;
SET
n3247_22_lab4=> grant delete on foreign_students to user4;
GRANT
n3247_22_lab4=> \dp foreign_students;
                                  Access privileges
 Schema |       Name        | Type  |     Access privileges     | Column privileges | Policies
--------+-------------------+-------+---------------------------+-------------------+----------
 public | foreign_students | table | postgres=arwdDxt/postgres+|                   |
        |                   |       | user1=arwdDxt/postgres   +|                   |
        |                   |       | user2=arw/postgres       +|                   |
        |                   |       | user3=d*/postgres        +|                   |
        |                   |       | user4=d/user3             |                   |
(1 row)
```

Рисунок 6 – Предоставим право от пользователя user3 пользователю user4

1.3.4 Задание 4

Отменим все предоставленные права.

```
n3247_22_lab4=# revoke all on foreign_students from user1;
REVOKE
n3247_22_lab4=# revoke insert, select, update on foreign_students from user2;
REVOKE
n3247_22_lab4=# revoke delete on foreign_students from user3 cascade;
REVOKE
n3247_22_lab4=# \dp foreign_students;
                                  Access privileges
 Schema |       Name        | Type  |     Access privileges     | Column privileges | Policies
--------+-------------------+-------+---------------------------+-------------------+----------
 public | foreign_students | table | postgres=arwdDxt/postgres |                   |
(1 row)
```

Рисунок 7 – Отметка представленных прав

1.3.5 Задание 5

Создадим подсхему для user1 и user2 с различным набором таблиц.

```
n3247_22_lab4=# create schema subschema_u1 authorization user1;
CREATE SCHEMA
n3247_22_lab4=# create schema subschema_u2 authorization user2;
CREATE SCHEMA
n3247_22_lab4=# create table subschema_u1.point1 (name text, point int);
CREATE TABLE
n3247_22_lab4=# create table subschema_u2.point2 (name text, point int);
CREATE TABLE
n3247_22_lab4=# \dn
        List of schemas
     Name     |       Owner
--------------+--------------------
 public       | pg_database_owner
 subschema_u1 | user1
 subschema_u2 | user2
(3 rows)
```

Рисунок 8 – Создание подсхемы для user1 и user2 с различным набором таблиц

⇒ *Пользователь* ***user1*** *из примера может видеть только таблицу из своей подсхемы.*

1.3.6 Задание 6

Создадим таблицу foreign_students2;

```
n3247_22_lab4=# create table foreign_students2 (id bigserial primary key, point int);
CREATE TABLE
```

Рисунок 9 – Создание таблицы foreign_students2

```
n3247_22_lab4=# insert into foreign_students2 (point) values (88), (75), (90), (100);
INSERT 0 4
n3247_22_lab4=# select * from foreign_students2;
 id | point
----+-------
  1 |    88
  2 |    75
  3 |    90
  4 |   100
(4 rows)
```

Рисунок 10 – Вставка оценки

Создадим представление students_info.

```
n3247_22_lab4=# create view student_info as select s.id, s.age, s.name, s2.point
n3247_22_lab4-# from foreign_students s, foreign_students2 s2 where s.id = s2.id;
CREATE VIEW
```

```
n3247_22_lab4=# select * from student_info;
 id | age |   name    | point
----+-----+-----------+-------
  1 |  22 | Doan      |    88
  2 |  20 | DamThuong |    75
  3 |  22 | Rikaono   |    90
  4 |  22 | Aibeili   |   100
(4 rows)
```

Рисунок 11 – Создание представления students_info.

Ограничьте доступ к созданному представлению для User1.

```
n3247_22_lab4=# revoke all on student_info from user1;
REVOKE
n3247_22_lab4=# set role user1;
SET
n3247_22_lab4⇒ select * from student_info;
ERROR:  permission denied for view student_info
```

Рисунок 12 – Проверка доступа к созданному представлению для User1

1.3.7 Задание 7

Создадим политику на основе пользователя user1 и протестируем ее.

```
n3247_22_lab4=# alter table foreign_students2 enable row level security;
ALTER TABLE
n3247_22_lab4=# create policy lvl_pol on foreign_students2 for select to user1 using (point ⩾ 90);
CREATE POLICY
n3247_22_lab4=# \dp foreign_students2;
                                    Access privileges
 Schema |       Name        | Type  |     Access privileges     | Column privileges |       Policies
--------+-------------------+-------+---------------------------+-------------------+----------------------
 public | foreign_students2 | table | postgres=arwdDxt/postgres+|                   | lvl_pol (r):        +
        |                   |       | user1=r/postgres          |                   |   (u): (point ⩾ 90)+
        |                   |       |                           |                   |   to: user1
(1 row)

n3247_22_lab4=# select* from foreign_students2;
 id | point
----+-------
  1 |    88
  2 |    75
  3 |    90
  4 |   100
(4 rows)

n3247_22_lab4=# set role user1;
SET
n3247_22_lab4⇒ select* from foreign_students2;
 id | point
----+-------
  3 |    90
  4 |   100
(2 rows)
```

Рисунок 13 – Создание политики

1.3.8 Задание 8

Создадим таблицы point_table (над которой мы будем выполнять операции) и log_table (где сохраняется информация об выполненных операциях):

```
n3247_22_lab4=# create table point_table (id int, point int);
CREATE TABLE
n3247_22_lab4=# create table log_table (query text, id int, point int);
CREATE TABLE
```

Рисунок 14 – Создание таблиц point_table и log_table

Создадим триггер для регистрации вставки, обновления и удаления над таблицей point_table в таблице log_table:

```
n3247_22_lab4=# CREATE OR REPLACE FUNCTION logfunc() RETURNS TRIGGER AS
$$
BEGIN
    IF TG_NAME = 'logfunc' THEN
        IF TG_OP IN ('INSERT', 'UPDATE') THEN
            RAISE NOTICE '% trigger function detected % sql query', TG_NAME, TG_OP;
            INSERT INTO log_table VALUES (TG_OP, NEW.id, NEW.point);
            RETURN NEW;
        END IF;

        IF TG_OP = 'DELETE' THEN
            RAISE NOTICE '% trigger function detected % sql query', TG_NAME, TG_OP;
            INSERT INTO log_table VALUES (TG_OP, OLD.id, OLD.point);
            RETURN OLD;
        END IF;
    END IF;
END;
$$
LANGUAGE plpgsql;
CREATE FUNCTION
```

Рисунок 15 – Создание триггера logfunc

После того протестируем:

```
n3247_22_lab4=# insert into point_table values (1, 84);
NOTICE:  logfunc trigger function detected INSERT sql query
INSERT 0 1
n3247_22_lab4=# insert into point_table values (2, 98);
NOTICE:  logfunc trigger function detected INSERT sql query
INSERT 0 1
n3247_22_lab4=# insert into point_table values (3, 75);
NOTICE:  logfunc trigger function detected INSERT sql query
INSERT 0 1
n3247_22_lab4=# update point_table set point = 99 where id = 2;
NOTICE:  logfunc trigger function detected UPDATE sql query
UPDATE 1
n3247_22_lab4=# select * from point_table;
 id | point
----+-------
  1 |    84
  3 |    75
  2 |    99
(3 rows)

n3247_22_lab4=# DELETE FROM
foreign_students     foreign_students2    information_schema.  log_table
n3247_22_lab4=# DELETE FROM point_table where id = 1;
NOTICE:  logfunc trigger function detected DELETE sql query
DELETE 1
n3247_22_lab4=# select * from point_table;
 id | point
----+-------
  3 |    75
  2 |    99
(2 rows)

n3247_22_lab4=# select * from log_table;
 query  | id | point
--------+----+-------
 INSERT |  1 |    84
 INSERT |  2 |    98
 INSERT |  3 |    75
 UPDATE |  2 |    99
 DELETE |  1 |    84
(5 rows)
```

Рисунок 16 – Протестирование триггера logfunc

Заключение

В ходе выполнения лабораторной работы был изучен теоретический материал по контролю доступа и системы аудита. Приобретенные знания были применены на практике в СУБД PostgreSQL.