Лабораторная работа №3

1. Найдите файл конфигурации postgresql.conf и измените параметры для активации сбора логов, указания директории для логов и установки уровня подробности логирования.

Файл находится по адресу:

C:\Program Files\PostgreSQL\16\data\postgresql.conf

# Включает механизм сбора логов

logging\_collector = on

# Указывает директорию для лог-файлов относительно каталога данных PostgreSQL

log\_directory = 'log'

# Устанавливает шаблон для имен файлов логов

log\_filename = 'postgresql-%Y-%m-%d\_%H%M%S.log'

# Указывает уровень подробности выводимых сообщений. Можно выбрать INFO, WARNING, ERROR и т. д.

log\_min\_messages = INFO

# Записывать ли в лог все SQL-команды

log\_statement = 'ddl'

#Директория для логов

C:\Program Files\PostgreSQL\16\data\log

# Логируем подключение/отключение пользователей

log\_connections = on

log\_disconnections = on

2. Задание на аудит и протоколирование в PostgreSQL:

Создайте правила аудита для отслеживания операций в базе данных, например, предположим, что вы должны расследовать следующие подозреваемые действия в базе данных:

1)Попытки доступа в базу данных в необычное время.

Просматриваются через лог файлы, в них указано время подключения к базе данных. За это отвечает параметр log\_connections=on.

1. Попытка доступа несуществующего пользователя. Отобразится в логах.

3)Отображение изменений в таблице.

1)CREATE SCHEMA Y

2)CREATE TABLE Y.X (

id SERIAL PRIMARY KEY,

name TEXT,

value INT

);

Создаём таблицу лог для изменений

CREATE TABLE public.audit\_log (

audit\_id SERIAL PRIMARY KEY,

operation TEXT, -- тип операции: INSERT, UPDATE, DELETE

schema\_name TEXT,

table\_name TEXT,

old\_data JSONB,

new\_data JSONB,

changed\_by TEXT,

changed\_at TIMESTAMP DEFAULT now()

);

3)Функция, которая пишет данные в эту таблицу

CREATE OR REPLACE FUNCTION public.audit\_table\_changes()

RETURNS trigger

LANGUAGE plpgsql

AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO public.audit\_log(

operation, schema\_name, table\_name, new\_data, changed\_by

)

VALUES (

'INSERT',

TG\_TABLE\_SCHEMA,

TG\_TABLE\_NAME,

to\_jsonb(NEW),

session\_user

);

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO public.audit\_log(

operation, schema\_name, table\_name, old\_data, new\_data, changed\_by

)

VALUES (

'UPDATE',

TG\_TABLE\_SCHEMA,

TG\_TABLE\_NAME,

to\_jsonb(OLD),

to\_jsonb(NEW),

session\_user

);

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO public.audit\_log(

operation, schema\_name, table\_name, old\_data, changed\_by

)

VALUES (

'DELETE',

TG\_TABLE\_SCHEMA,

TG\_TABLE\_NAME,

to\_jsonb(OLD),

session\_user

);

RETURN OLD;

END IF;

END;

$$;

5)Вешаем триггер на таблицу

CREATE TRIGGER trig\_audit\_X

AFTER INSERT OR UPDATE OR DELETE

ON Y.X

FOR EACH ROW

EXECUTE FUNCTION public.audit\_table\_changes();

6)Проверяем работу триггера

INSERT INTO Y.X (name, value) VALUES ('Test insert', 123);

UPDATE Y.X

SET value = 999

WHERE name = 'Test insert';

DELETE FROM Y.X

WHERE name = 'Test insert';

7)Проверяем лог

SELECT \*

FROM public.audit\_log

ORDER BY audit\_id;

4)Количество обращений

1)Создаём таблицу Z

CREATE TABLE Z (

id SERIAL PRIMARY KEY,

info TEXT

);

2)Создаём таблицу счётчик обращений

CREATE TABLE Z\_access\_count (

id SERIAL PRIMARY KEY,

operation\_date TIMESTAMP DEFAULT now(),

operation\_type TEXT,

comment TEXT

);

3)Создаём функцию для учёта обращений

CREATE OR REPLACE FUNCTION fn\_incr\_Z\_access()

RETURNS trigger

LANGUAGE plpgsql

AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO Z\_access\_count(operation\_type, comment)

VALUES ('INSERT', 'Вставлена новая строка в Z');

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO Z\_access\_count(operation\_type, comment)

VALUES ('UPDATE', 'Обновлена строка в Z');

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO Z\_access\_count(operation\_type, comment)

VALUES ('DELETE', 'Удалена строка из Z');

RETURN OLD;

END IF;

END;

$$;

Вещаем триггер на таблицу Z

CREATE TRIGGER trig\_count\_Z

AFTER INSERT OR UPDATE OR DELETE

ON Z

FOR EACH ROW

EXECUTE FUNCTION fn\_incr\_Z\_access();

5)Тестируем

INSERT INTO Z (info) VALUES ('Hello!');

SELECT \* FROM Z\_access\_count;

Обновление

UPDATE Z

SET info = 'Hello world!'

WHERE id = 1;

SELECT \* FROM Z\_access\_count;

Удаление

DELETE FROM Z

WHERE id = 1;

SELECT \* FROM Z\_access\_count;