Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №5 з дисципліни «Бази даних»

«Основи програмування з

використанням мови SQL. Збережені процедури. Курсори. Створення, програмування та керування тригерами.»

Варіант 10

Виконав студент ІП-13, Замковий Дмитро Володимирович

(шифр, прізвище, ім'я, по батькові)

Перевірив Марченко Олена Іванівна\_\_\_\_\_\_\_\_\_\_\_\_\_

( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота 5**

**Основи програмування з використанням мови SQL. Збережені процедури. Курсори. Створення, програмування та керування тригерами.**

**Мета заняття:** Вивчити правила побудови ідентифікаторів, правила визначення змінних та типів. Визначити правила роботи з циклами та умовними конструкціями, роботу зі змінними типу Table. Вивчити синтаксис та семантику функцій та збережених процедур, способів їх ідентифікації, методів визначення та специфікації параметрів та повертаємих значень, виклик функцій та збережених процедур. Застосування команд для створення, зміни та видалення як скалярних, такі табличних функцій, збережених процедур. Вивчити призначення та типи курсорів, синтаксис та семантику команд мови SQL для створення курсорів, вибірки даних з курсорів, зміни даних із застосуванням курсорів. Вивчити призначення та типи тригерів, умов їх активації, синтаксису та семантики для їх створення, модифікації, перейменування, програмування та видалення.

**Завдання:**

При виконанні лабораторної роботи необхідно виконати наступні дії:

1. Збережені процедури:

запит для створення тимчасової таблиці через змінну типу TABLE;

запит з використанням умовної конструкції IF;

запит з використанням циклу WHILE;

створення процедури без параметрів;

створення процедури з вхідним параметром;

створення процедури з вхідним параметром та RETURN;

створення процедури оновлення даних в деякій таблиці БД;

створення процедури, в котрій робиться вибірка даних.

1. Функції:

створити функцію, котра повертає деяке скалярне значення;

створити функцію, котра повертає таблицю з динамічним набором стовпців;

створити функцію, котра повертає таблицю заданої структури.

1. Робота з курсорами:

створити курсор;

відкрити курсор;

вибірка даних, робота з курсорами.

1. Робота з тригерами:

створити тригер, котрий буде спрацьовувати при видаленні даних;

створити тригер, котрий буде спрацьовувати при модифікації даних;

створити тригер, котрий буде спрацьовувати при додаванні даних

**Скрипт:**

-- Процедура з тимчасовою таблицею

CREATE OR REPLACE PROCEDURE create\_table(IN lim INTEGER DEFAULT(10))

AS $$

    BEGIN

        DROP TABLE IF EXISTS tmp\_table;

        CREATE TEMP TABLE tmp\_table AS

            SELECT \*

            FROM branch

            LIMIT lim;

    END

$$ LANGUAGE plpgsql;

CALL create\_table();

SELECT \* FROM tmp\_table;

-- Процедура з використанням умовної конструкції IF

CREATE OR REPLACE PROCEDURE if\_p(IN tariff INTEGER)

AS $$

    BEGIN

        IF tariff > 450 THEN

            raise notice 'Високий';

        ELSE

            raise notice 'Низький';

        END IF;

    END

$$ LANGUAGE plpgsql;

CALL if\_p(500);

CALL if\_p(200);

-- Процедура з використанням циклу WHILE

CREATE OR REPLACE PROCEDURE while\_p(IN num INTEGER)

AS $$

    DECLARE

        i INTEGER := 0;

    BEGIN

        WHILE i <= num LOOP

            raise notice '%', i;

            i := i + 1;

        END LOOP;

    END;

$$ LANGUAGE plpgsql;

CALL while\_p(7);

-- Процедура без параметрів

CREATE OR REPLACE PROCEDURE hello\_word()

AS $$

    BEGIN

    raise notice 'Hello world!';

    END;

$$ LANGUAGE plpgsql;

CALL hello\_word();

-- Процедура з вхідним параметром

CREATE OR REPLACE PROCEDURE hello\_user(in user\_name VARCHAR(20))

AS $$

    BEGIN

    raise notice 'Hello, %!', user\_name;

    END;

$$ LANGUAGE plpgsql;

CALL hello\_user('Sam');

-- Процедура з вхідним параметром та RETURN

CREATE OR REPLACE PROCEDURE sum\_p (INOUT res INTEGER DEFAULT NULL)

AS $$

    BEGIN

        SELECT SUM(employee.tariff)

        FROM employee

        INTO res;

    END;

$$ LANGUAGE plpgsql;

CALL sum\_p();

-- Процедура оновлення даних в деякій таблиці БД

CREATE OR REPLACE PROCEDURE refresh\_employee(IN employee\_id INTEGER, IN new\_tariff INTEGER)

AS $$

    BEGIN

        UPDATE employee

        SET tariff = new\_tariff

        WHERE id = employee\_id;

    END;

$$ LANGUAGE plpgsql;

SELECT \* FROM employee WHERE employee."id"=1;

CALL refresh\_employee(1, 200);

SELECT \* FROM employee WHERE employee."id"=1;

-- Процедура в котрій робиться вибірка даних

CREATE OR REPLACE PROCEDURE select\_employee()

AS $$

    BEGIN

        DROP TABLE IF EXISTS tmp\_employee;

        CREATE TEMP TABLE tmp\_employee AS

        SELECT employee."id", employee.first\_name, employee.phone\_number, employee.tariff

        FROM employee;

    END;

$$ LANGUAGE plpgsql;

CALL select\_employee();

SELECT \* FROM tmp\_employee;

-- Функція котра повертає деяке скалярне значення

CREATE OR REPLACE FUNCTION sum\_f() RETURNS NUMERIC

AS $$

    BEGIN

        RETURN(SELECT SUM(employee.tariff) FROM employee);

    END;

$$ LANGUAGE plpgsql;

SELECT sum\_f();

-- Функція котра повертає таблицю з динамічним набором стовпців

CREATE OR REPLACE FUNCTION dynamic\_f(IN tbl\_type anyelement) RETURNS SETOF anyelement

AS $$

DECLARE

    t\_name varchar := pg\_typeof(tbl\_type);

BEGIN

    return query

        execute format('select \* from %s;', t\_name);

END;

$$ LANGUAGE plpgsql;

select \* from dynamic\_f(null::employee);

-- Функція котра повертає таблицю заданої структури

CREATE OR REPLACE FUNCTION select\_employees()

RETURNS TABLE (

    id INTEGER,

    first\_name VARCHAR(40),

    phone\_number BIGINT,

    tariff INTEGER

)

AS $$

    BEGIN

        RETURN query SELECT employee."id", employee.first\_name, employee.phone\_number, employee.tariff FROM employee ORDER BY id;

    END;

$$ LANGUAGE plpgsql;

SELECT \* FROM select\_employees();

-- Робота з курсорами

DO $$

    DECLARE

        cur CURSOR FOR SELECT \* FROM employee;

        emp employee;

    BEGIN

        OPEN cur;

        LOOP

            FETCH cur INTO emp;

            EXIT WHEN NOT FOUND;

            raise notice '{id: %, first\_name: %, tariff: %}', emp.id, emp.first\_name, emp.tariff;

        END LOOP;

    CLOSE cur;

END;

$$;

-- Тригер котрий буде спрацьовувати при видаленні даних

CREATE OR REPLACE FUNCTION t\_dell\_f() RETURNS TRIGGER

AS $$

    BEGIN

        raise notice '

            Видалено працівника

            ПІБ: % %

            Тариф: %

            ІД філії: %

        ', OLD.first\_name, OLD.lasr\_name, OLD.tariff, OLD.id\_branch;

        RETURN NEW;

    END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER t\_dell

    AFTER DELETE

    ON employee

    FOR EACH ROW

EXECUTE FUNCTION t\_dell\_f();

DELETE FROM employee WHERE employee."id" = 22;

-- Тригер котрий буде спрацьовувати при модифікації даних

CREATE OR REPLACE FUNCTION t\_upd\_f() RETURNS TRIGGER

AS $$

    BEGIN

        raise notice '

            Змінено тариф працівника

            Прізвище: %.

            Старий тариф: %.

            Новий тариф: %

        ', NEW.first\_name, OLD.tariff, NEW.tariff;

        RETURN NEW;

    END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER t\_upd

    AFTER UPDATE

    ON employee

    FOR EACH ROW

    WHEN (OLD.tariff != NEW.tariff)

EXECUTE FUNCTION t\_upd\_f();

UPDATE employee SET tariff = 500 WHERE id = 22;

-- Тригер котрий буде спрацьовувати при додаванні даних

CREATE OR REPLACE FUNCTION t\_insert\_f() RETURNS TRIGGER

AS $$

    BEGIN

        raise notice '

            Додано нового працівника

            ПІБ: % %.

            Номер телефону: +%.

            Тариф: %

        ', NEW.first\_name, NEW.lasr\_name, NEW.phone\_number, NEW.tariff;

        RETURN NEW;

    END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER t\_insert

    AFTER INSERT

    ON employee

    FOR EACH ROW

EXECUTE FUNCTION t\_insert\_f();

INSERT INTO employee VALUES (22, 'sfdfvd', 'sdvdcsd', NULL, 2, 380958745236, 300, 2);

**Висновок:**

В ході даної лабораторної роботи я вивчив правила побудови ідентифікаторів, правила визначення змінних та типів, визначив правила роботи з циклами та умовними конструкціями, роботу зі змінними типу Table. Вивчив синтаксис та семантику функцій та збережених процедур, способів їх ідентифікації, методів визначення та специфікації параметрів та повертаємих значень, виклик функцій та збережених процедур. Застосував команди для створення, зміни та видалення як скалярних, такі табличних функцій, збережених процедур. Вивчв призначення та типи курсорів, синтаксис та семантику команд мови SQL для створення курсорів, вибірки даних з курсорів, зміни даних із застосуванням курсорів. Вивчив призначення та типи тригерів, умов їх активації, синтаксису та семантики для їх створення, модифікації, перейменування, програмування та видалення.