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)

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:
    a. запит для створення тимчасової таблиці через змінну типу TABLE;
    b. запит з використанням умовної конструкції IF;
    c. запит з використанням циклу WHILE;
    d. створення процедури без параметрів;
    e. створення процедури з вхідним параметром;
    f. створення процедури з вхідним параметром та RETURN;
    g. створення процедури оновлення даних в деякій таблиці БД;
    h. створення процедури, в котрій робиться вибірка даних.
2) Функції:
    a. створити функцію, котра повертає деяке скалярне значення;
    b. створити функцію, котра повертає таблицю з динамічним набором стовпців;
    c. створити функцію, котра повертає таблицю заданої структури.
3) Робота з курсорами:
    a. створити курсор;
    b. відкрити курсор;
    c. вибірка даних, робота з курсорами.
4) Робота з тригерами:
    a. створити тригер, котрий буде спрацьовувати при видаленні даних;
    b. створити тригер, котрий буде спрацьовувати при модифікації даних;
    c. 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.