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 1

Виконав студент ІП-13 Ал Хадам Мурат Резгович
(шифр, прізвище, ім'я, по батькові)

Перевірив Марченко Олена Іванівна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №5

Варіант 1

Мета:

- 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, так і 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.

Постановка задачі лабораторної робота № 5

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.

Виконання завдання

Створені запити:

task1

```
USE `db-labs`;

# 1 a
DROP PROCEDURE IF EXISTS temp_five_patients;
DELIMITER $$
CREATE PROCEDURE temp_five_patients()
BEGIN
    DROP TABLE IF EXISTS temp;
    CREATE TEMPORARY TABLE temp AS (SELECT *
                                    FROM patient_info
                                    LIMIT 5);
    SELECT *
    FROM temp;
END $$
DELIMITER ;
CALL temp_five_patients();


# 1 b
DROP PROCEDURE IF EXISTS compare_height;
DELIMITER $$
CREATE PROCEDURE compare_height(IN height_to_compare INT)
BEGIN
    SELECT full_name,
           IF(height > height_to_compare,
              CONCAT('ВИЩЕ', ' ', height_to_compare),
              CONCAT('НИЖЧЕ', ' ', height_to_compare)
               ) AS height_comparing
    FROM patient_info;
END $$
DELIMITER ;
CALL compare_height(185);


# 1 c
DROP PROCEDURE IF EXISTS do_while;
DELIMITER $$
CREATE PROCEDURE do_while()
BEGIN
    DECLARE var INT DEFAULT 5;

    WHILE var > 0
        DO
            SET var = var - 1;
        END WHILE;

    SELECT var;
END $$
DELIMITER ;
CALL do_while();


# 1 d
DROP PROCEDURE IF EXISTS get_beneficiaries;
DELIMITER $$
CREATE PROCEDURE get_beneficiaries()
BEGIN
    SELECT *
    FROM patients
    WHERE is_beneficiary;
END $$
DELIMITER ;
```

```
CALL get_beneficiaries();

# 1 e
DROP PROCEDURE IF EXISTS get_patient_by_gender;
DELIMITER $$
CREATE PROCEDURE get_patient_by_gender(
    IN gen VARCHAR(20)
)
BEGIN
    SELECT *
    FROM patients
    WHERE gender = gen;
END $$
DELIMITER ;
CALL get_patient_by_gender('чоловік');

# 1 f
DROP PROCEDURE IF EXISTS get_amount_by_bmi_limit;
DELIMITER $$
CREATE PROCEDURE get_amount_by_bmi_limit(
    IN bmi_max INT,
    OUT amount INT
)
BEGIN
    SELECT COUNT(*)
    FROM patient_info
    WHERE bmi < bmi_max
    INTO amount;

    SELECT amount;
END $$
DELIMITER ;
CALL get_amount_by_bmi_limit(20, @amount);

# 1 g
DROP PROCEDURE IF EXISTS update_discount;
DELIMITER $$
CREATE PROCEDURE update_discount()
BEGIN
    UPDATE discounts
    SET percentage = percentage * 1.1
    WHERE percentage < 10;
END $$
DELIMITER ;
CALL update_discount();

# 1 h
DROP PROCEDURE IF EXISTS check_admins;
DELIMITER $$
CREATE PROCEDURE check_admins()
BEGIN
    SELECT *
    FROM admins
    WHERE regular_patients > 7;
END $$
DELIMITER ;
CALL check_admins();
```

task2

```sql
USE `db-labs`;

# 2 a
DROP FUNCTION IF EXISTS get_high_qual_doctors;
DELIMITER $$
CREATE FUNCTION get_high_qual_doctors() RETURNS INT
    DETERMINISTIC
BEGIN
    DECLARE amount INT;
    SELECT COUNT(*)
    FROM doctors
    WHERE qualification = 'вища'
    INTO amount;
    RETURN amount;
END $$
DELIMITER ;
SELECT get_high_qual_doctors() AS high_qual;

# 2 b (stored procedure for this task)
DROP PROCEDURE IF EXISTS get_by_spec;
DELIMITER $$
CREATE PROCEDURE get_by_spec(
    IN spec VARCHAR(20)
)
BEGIN
    SELECT *
    FROM doctors
    WHERE specialization = spec;
END $$
DELIMITER ;
CALL get_by_spec('кардіоревматолог');

# 2 c
DELIMITER $$
DROP FUNCTION IF EXISTS get_table;
CREATE FUNCTION get_table(t_name VARCHAR(20)) RETURNS VARCHAR(2048)
    DETERMINISTIC
BEGIN
    DECLARE result VARCHAR(2048);
    DECLARE temp1, temp2 VARCHAR(1024);
    SELECT GROUP_CONCAT(COLUMN_NAME)
    INTO temp1
    FROM information_schema.COLUMNS
    WHERE TABLE_SCHEMA = DATABASE()
      AND TABLE_NAME = t_name;

    SELECT GROUP_CONCAT(PRIVILEGES)
    INTO temp2
    FROM information_schema.COLUMNS
    WHERE TABLE_SCHEMA = DATABASE()
      AND TABLE_NAME = t_name;

    SET result = CONCAT(temp1, temp2);
    RETURN result;
END $$
DELIMITER ;
SELECT get_table('patients');
```

task3/4.sql

```sql
USE `db-labs`;

# 3
DROP PROCEDURE IF EXISTS cur_demo;
DELIMITER $$
CREATE PROCEDURE cur_demo()
BEGIN
    DECLARE done TINYINT DEFAULT FALSE;
    DECLARE cur_spec VARCHAR(40);

    DECLARE cur CURSOR FOR
        SELECT specialization FROM doctors;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop:
    LOOP
        FETCH cur INTO cur_spec;
        IF done THEN
            LEAVE read_loop;
        END IF;
    END LOOP;

    SELECT cur_spec;
END $$
DELIMITER ;
CALL cur_demo();

# 4 a
DROP TRIGGER IF EXISTS trigger_delete;
DELIMITER $$
CREATE TRIGGER trigger_delete
    BEFORE DELETE
    ON admins
    FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*)
        FROM admins
        WHERE shift number = OLD.shift number) = 3 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'ВИ НЕ МОЖЕТЕ ВИДАЛИТИ АДМІНА, ТАК ЯК
                                ЗАЛИШЕТЬСЯ ЛИШЕ 2 АДМІНА НА ЗМІНІ';
    END IF;
END $$
DELIMITER ;
# -------------
DELETE
FROM admins
WHERE admin_id = 11;

# 4 b
DROP TRIGGER IF EXISTS trigger_update;
DELIMITER $$
CREATE TRIGGER trigger_update
    BEFORE UPDATE
    ON cards
    FOR EACH ROW
BEGIN
    IF OLD.height != NEW.height OR
       OLD.weight != NEW.weight
    THEN
        IF ROUND(NEW.weight / POW((NEW.height / 100), 2), 2) > 24.9 OR
           ROUND(NEW.weight / POW((NEW.height / 100), 2), 2) < 18.5
```

```
        THEN
            SET NEW.medical_group = 'спеціальна';
        END IF;
    END IF;
END $$
DELIMITER ;
# ---------------
UPDATE cards
SET height = 10
WHERE patient_id = 27;

# 4 c
DROP TRIGGER IF EXISTS trigger_insert;
DELIMITER $$
CREATE TRIGGER trigger_insert
    BEFORE INSERT
    ON admins
    FOR EACH ROW
BEGIN
    IF (SELECT COUNT(*)
        FROM admins
        WHERE shift_number = NEW.shift_number) > 1 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'ВИ НЕ МОЖЕТЕ ДОДАТИ АДМІНА НА ЦЮ ЗМІНУ, ТАК ЯК
                                ВЖЕ МАЄТЬСЯ 2 АБО БІЛЬШЕ АДМІНІВ НА ЗМІНІ';
    END IF;
END $$
DELIMITER ;
# ---------------
INSERT INTO admins (first_name, last_name, regular_patients, shift_number)
VALUES ('TEST-INSERT', 'TEST-INSERT', 0, 101);
```

Висновок:

На даній лабораторній роботі я попрактикувався у створенні та використанні збережених процедур та функцій на мові SQL. Вивчив можливості створення процедур з вхідними та вихідними параметрами. Також, навчився працювати з курсорами та тригерами.