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 19

Виконав студент ІП-13 Нещерет Віталій Олександрович
(шифр, прізвище, ім'я, по батькові)

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

Лабораторна робота №5

Варіант 19

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

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

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:

```
USE exchequer;

DROP PROCEDURE IF EXISTS create_table;
CREATE PROCEDURE create_table()
BEGIN
    DROP TABLE IF EXISTS temp_table;
    CREATE TEMPORARY TABLE temp_table AS
        SELECT *
        FROM `кошторис`
        LIMIT 15;
END;
CALL create_table();

DROP PROCEDURE IF EXISTS procedure_with_if;
CREATE PROCEDURE procedure_with_if()
BEGIN
    SELECT `кошторис id`,
           `ліміт витрат`,
           IF(`ліміт витрат` > 5000000, 'Великий бюджет',
               'Маленький бюджет')
               AS 'розмір бюджету'
    FROM `кошторис`
    ORDER BY `розмір бюджету`;
END;
CALL procedure_with_if();

DROP PROCEDURE IF EXISTS procedure_with_while;
CREATE PROCEDURE procedure_with_while()
BEGIN
    DECLARE counter INT DEFAULT 5;

    WHILE counter > 0 DO
        SELECT counter;
        SET counter = counter - 1;
    END WHILE;
END;
CALL procedure_with_while();

DROP PROCEDURE IF EXISTS procedure_with_while;
CREATE PROCEDURE procedure_with_while()
BEGIN
    DECLARE counter INT DEFAULT 5;

    WHILE counter > 0 DO
        SELECT counter;
        SET counter = counter - 1;
    END WHILE;
END;
CALL procedure_with_while();

DROP PROCEDURE IF EXISTS procedure_without_params;
CREATE PROCEDURE procedure_without_params()
BEGIN
    SELECT *
    FROM кекв;
END;
CALL procedure_without_params();

DROP PROCEDURE IF EXISTS procedure_with_params;
CREATE PROCEDURE procedure_with_params(
    IN max_sum INT
```

```
)
BEGIN
    SELECT *
    FROM кошторис
    WHERE `витрачена сума` < max_sum;
END;
CALL procedure_with_params(1000000);

DROP PROCEDURE IF EXISTS procedure_with_in_and_out;
CREATE PROCEDURE procedure_with_in_and_out(
    IN max_sum INT,
    OUT res INT
)
BEGIN
    SELECT COUNT(*)
    FROM кошторис
    WHERE `витрачена сума` < max_sum
    INTO res;
    SELECT res;
END;
CALL procedure_with_in_and_out(1000000, @res);

DROP PROCEDURE IF EXISTS update_table;
CREATE PROCEDURE update_table()
BEGIN
    UPDATE `кошторис`
    SET `витрачена сума` = `витрачена сума` + 100000
    WHERE `витрачена сума` < 1000000;
END;
CALL update_table();

DROP PROCEDURE IF EXISTS select_procedure;
CREATE PROCEDURE select_procedure()
BEGIN
    SELECT *
    FROM `кошторис`
    WHERE `витрачена сума` < 1000000;
END;
CALL select_procedure();

DROP FUNCTION IF EXISTS function_scalar;
CREATE  FUNCTION  function_scalar() RETURNS INT
READS SQL DATA
BEGIN
    DECLARE res INT;
    SELECT COUNT(*)
    FROM `кошторис`
    WHERE `витрачена сума` < 1000000
    INTO res;
    RETURN res;
END;
SELECT function_scalar();

-- Function in MySQL cannot return a table. So, I use stored procedure for this
task.
DROP PROCEDURE IF EXISTS procedure_table_with_dynamic_columns;
CREATE PROCEDURE procedure_table_with_dynamic_columns(
    IN magic_number INT
)
BEGIN
    IF magic_number = 1 THEN
        SELECT `кекв id`, `витрачена сума`
        FROM `кекв`;
    ELSE
        SELECT *
        FROM `кекв`;
    END IF;
```

```
END;
CALL procedure_table_with_dynamic_columns(1);

DROP PROCEDURE IF EXISTS work_with_cursor;
CREATE PROCEDURE work_with_cursor(
)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE name VARCHAR(255);
    DECLARE cur CURSOR FOR
        SELECT назва
        FROM `комерційний банк`;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO name;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
        SELECT name;
    END LOOP;
END;
CALL work_with_cursor();

DROP TRIGGER IF EXISTS trigger_delete;
CREATE TRIGGER trigger_delete BEFORE DELETE ON `кошторис` FOR EACH ROW BEGIN
    IF OLD.`витрачена сума` > 1000000 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Ви не можете видалити запис, в якому витрачена сума
більша за 1000000';
    END IF;
END;

DROP TRIGGER IF EXISTS trigger_update;
CREATE TRIGGER trigger_update BEFORE UPDATE ON `кошторис` FOR EACH ROW BEGIN
    IF NEW.`витрачена сума` > 1000000 THEN
        SET NEW.`витрачена сума` = 1000000;
    END IF;
END;

DROP TRIGGER IF EXISTS trigger_insert;
CREATE TRIGGER trigger_insert BEFORE INSERT ON `кошторис` FOR EACH ROW BEGIN
    IF NEW.`витрачена сума` > 1000000 THEN
        SET NEW.`витрачена сума` = 1000000;
    END IF;
END;
```

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.