ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №9

Тема: «Збережені процедури СКБД PostgreSQL».

Виконала:
Ст. *Пелещак Вероніка*
ПМІ-35с

2025

Тема: *«Збережені процедури СКБД PostgreSQL».*

Мета роботи: Вивчення поняття збережених процедур СКБД PostgreSQL v.11.

Завдання лабораторної роботи:

1. Розробити три збережені процедури відповідно
до власних потреб роботи зі створюваною базою даних.

2. У збережених процедурах необхідно використати:

- оголошення вхідних та вихідних параметрів функції, псевдоніми, змінні та константи;
- присвоювання змінній типу RECORD та команди PERFORM, STRICT або EXECUTE на вибір;
- керуючі структури (IF/CASE/FOR … на вибір);
- обов'язково команди обробки помилок EXCEPTION та виведення повідомлень RAISE;
- обов'язково використати курсор.

Теоретична частина:

1. Огляд PL/pgSQL

- PL/pgSQL – процедурна мова PostgreSQL, яка розширює можливості SQL.
- Дозволяє використовувати змінні, курсори, керуючі конструкції, обробку помилок.
- Переваги: більша гнучкість, скорочення коду, повторне використання, підвищення продуктивності.

2. Використані можливості

- **Вхідні параметри (IN)** – для передачі значень у процедуру.
- **Змінні** (TEXT, RECORD) – для збереження проміжних результатів.
- **PERFORM** – виконання SQL-запиту без повернення значення.

- **STRICT** – для перевірки, що SELECT повертає рівно один рядок.
- **IF/CASE** – для перевірки умов.
- **LOOP + CURSOR** – для послідовної обробки кількох рядків.
- **RAISE NOTICE** – для виводу інформації користувачу.
- **RAISE EXCEPTION** – для генерації помилок.
- **JOIN** – для отримання повної інформації з кількох таблиць.

Хід роботи:

1. Процедура для перегляду всіх багів певного користувача:

```
CREATE OR REPLACE PROCEDURE get_user_bugs_proc (
    IN p_user_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    bug_rec RECORD;
    bug_cur CURSOR FOR
        SELECT b.bug_id,
               b.bug_name,
               b.status,
               b.priority
        FROM bugs b
        WHERE b.created_user_id = p_user_id;
BEGIN
    PERFORM 1 FROM users u WHERE u.user_id = p_user_id;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'User with id % does not exist', p_user_id;
    END IF;

    OPEN bug_cur;
    LOOP
        FETCH bug_cur INTO bug_rec;
        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Bug id: %, Name: %, Status: %, Priority: %',
            bug_rec.bug_id, bug_rec.bug_name, bug_rec.status, bug_rec.priority;
    END LOOP;
    CLOSE bug_cur;

END;
$$;

CALL get_user_bugs_proc(3);
```

Data Output | Messages | Notifications

```
NOTICE:  Bug id: 1, Name: Нерівний розмір шрифта, Status: New, Priority: Low
NOTICE:  Bug id: 2, Name: Помилка при запуску програми, Status: In Progress, Priority: Critical
NOTICE:  Bug id: 7, Name: Некоректний пошук, Status: Fixed, Priority: Medium
CALL
```

2. Процедура, яка призначає баг конкретному розробнику:

```sql
CREATE OR REPLACE PROCEDURE assign_bug_proc(
    IN p_bug_id INT,
    IN p_user_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_role TEXT;
    v_bug_name TEXT;
    p_result TEXT;
BEGIN
    SELECT user_role INTO STRICT v_role
    FROM users
    WHERE user_id = p_user_id;

    IF LOWER(v_role) <> 'developer' THEN
        RAISE EXCEPTION 'User % is not developer (role: %)', p_user_id, v_role;
    END IF;

    SELECT bug_name INTO STRICT v_bug_name
    FROM bugs
    WHERE bug_id = p_bug_id;

    UPDATE bugs
    SET assigned_user_id = p_user_id
    WHERE bug_id = p_bug_id;

    p_result := format('Bug "%s" (ID %s) successfully assigned to user %s',
                       v_bug_name, p_bug_id, p_user_id);
    RAISE NOTICE '%', p_result;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_result := format('User or bug not found (bug_id=%s / user_id=%s)', p_bug_id, p_user_id);
        RAISE NOTICE '%', p_result;
    WHEN OTHERS THEN
        RAISE EXCEPTION 'Error assigning bug % to user %: %', p_bug_id, p_user_id, SQLERRM;
END;
$$;
```

```sql
CALL assign_bug_proc(3, 2);
```

Data Output | Messages | Notifications

```
NOTICE:  Bug "Несправність інтерфейсу" (ID 3) successfully assigned to user 2
CALL

Query returned successfully in 40 msec.
```

3. Процедура для перегляду історії змін по конкретному багу:

```sql
CREATE OR REPLACE PROCEDURE show_bug_history_proc(
    IN p_bug_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    hist_rec RECORD;
    cur_hist CURSOR FOR
        SELECT  h.corrected_id, h.date,
                (u.first_name || ' ' || u.last_name) AS user_name,
                h.corrected_coment
        FROM history_corrections h
        JOIN users u ON u.user_id = h.user_id
        WHERE h.bug_id = p_bug_id
        ORDER BY h.date;
BEGIN
    PERFORM 1 FROM bugs b WHERE b.bug_id = p_bug_id;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Bug with id % does not exist', p_bug_id;
    END IF;

    OPEN cur_hist;
    LOOP
        FETCH cur_hist INTO hist_rec;
        EXIT WHEN NOT FOUND;

        CASE
            WHEN hist_rec.corrected_coment ILIKE '%покращено%' THEN
                RAISE NOTICE 'Correction ID=% | Date=% | By=% | Comment: %',
                    hist_rec.corrected_id, hist_rec.date, hist_rec.user_name, hist_rec.corrected_coment;
            ELSE
                RAISE NOTICE 'Update ID=% | Date=% | By=% | Comment: %',
                    hist_rec.corrected_id, hist_rec.date, hist_rec.user_name, hist_rec.corrected_coment;
        END CASE;
    END LOOP;
    CLOSE cur_hist;

EXCEPTION
    WHEN OTHERS THEN
        RAISE EXCEPTION 'Error getting history for bug %: %', p_bug_id, SQLERRM;
END;
$$;
```

```sql

CALL show_bug_history_proc(3);

```

Data Output | Messages | Notifications

```
NOTICE:  Update ID=5 | Date=2025-03-01 | By=Maria Kozak | Comment: Шрифт було адаптовано для різних розмірів екранів.
CALL

Query returned successfully in 99 msec.
```

4. Приклади обробки виключень:

```
126
127  CALL get_user_bugs_proc(999);
128
129
130
131
```

Data Output | Messages | Notifications

```
ERROR:  User with id 999 does not exist
CONTEXT:  PL/pgSQL function get_user_bugs_proc(integer) line 14 at RAISE

SQL state: P0001
```

```
129
130  CALL assign_bug_proc(3, 11);
131
132
```

Data Output | Messages | Notifications

```
NOTICE:  User or bug not found (bug_id=3 / user_id=11)
CALL

Query returned successfully in 38 msec.
```

```
132
133  CALL assign_bug_proc(2, 1);
134
135
```

Data Output | Messages | Notifications

```
ERROR:  Error assigning bug 2 to user 1: User 1 is not developer (role: Tester)
CONTEXT:  PL/pgSQL function assign_bug_proc(integer,integer) line 32 at RAISE

SQL state: P0001
```

```
135
136  CALL show_bug_history_proc(25);
137
138
```

Data Output | Messages | Notifications

```
ERROR:  Error getting history for bug 25: Bug with id 25 does not exist
CONTEXT:  PL/pgSQL function show_bug_history_proc(integer) line 36 at RAISE

SQL state: P0001
```

Висновок: У результаті виконання лабораторної роботи були створені три збережені процедури мовою PL/pgSQL, які реалізують практичні задачі роботи з базою даних. У процесі розробки використано курсори, змінні типу RECORD, конструкції IF та CASE, команди PERFORM і STRICT, а також механізм обробки виключень EXCEPTION із повідомленнями RAISE.

Ці процедури дозволяють отримувати список багів користувача, призначати баги розробникам і переглядати історію їх виправлень. Таким чином було закріплено знання з PL/pgSQL і продемонстровано його можливості для реалізації бізнес-логіки в базі даних.