Лабораторна робота № 5. 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.

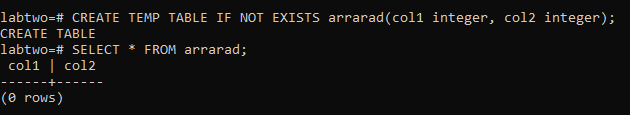
− Вивчити призначення та типи тригерів, умов їх активації, синтаксису та семантики для їх створення, модифікації, перейменування, програмування та видалення.

1) Збережені процедури:

a. запит для створення тимчасової таблиці через змінну типу TABLE;

Створимо тимчасову таблицю з двох колонок, в яких зберігаються цілі числа:

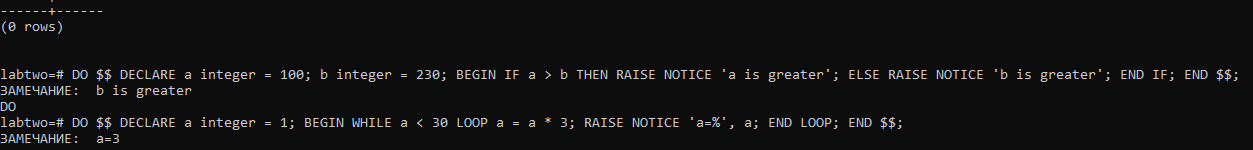
CREATE TEMP TABLE IF NOT EXISTS arrarad(col1 integer, col2 integer);



b. запит з використанням умовної конструкції IF;

Створимо запит, котрий порівняє a (100) і b (230) і повідомить, яке число більше:

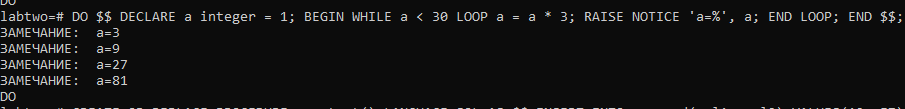
DO $$ DECLARE a integer = 100; b integer = 230; BEGIN IF a > b THEN RAISE NOTICE 'a is greater'; ELSE RAISE NOTICE 'b is greater'; END IF; END $$;



c. запит з використанням циклу WHILE;

Створимо запит, який буде виводити число, рівне степіні трійки, до тих пір, поки число не перевищить 30:

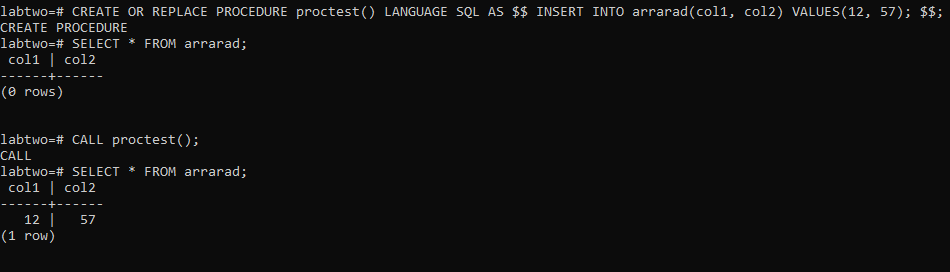
DO $$ DECLARE a integer = 1; BEGIN WHILE a < 30 LOOP a = a \* 3; RAISE NOTICE 'a=%', a; END LOOP; END $$;



d. створення процедури без параметрів;

Створимо процедуру, що додасть у таблицю з а. рядок 12, 57:

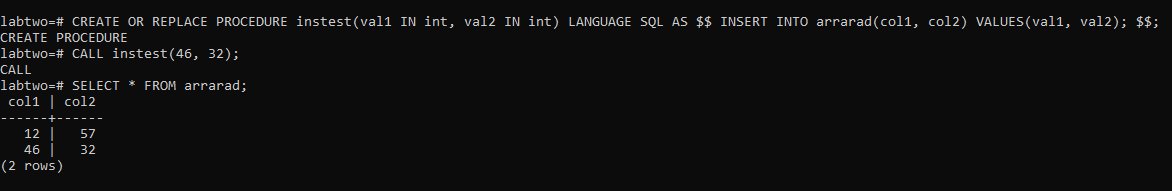
CREATE OR REPLACE PROCEDURE proctest() LANGUAGE SQL AS $$ INSERT INTO arrarad(col1, col2) VALUES(12, 57); $$;



e. створення процедури з вхідним параметром;

Теж саме, що в d., але числа задаємо самі:

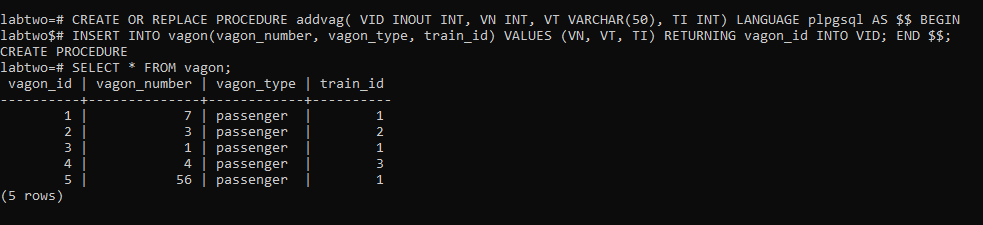
CREATE OR REPLACE PROCEDURE instest(val1 IN int, val2 IN int) LANGUAGE SQL AS $$ INSERT INTO arrarad(col1, col2) VALUES(val1, val2); $$;

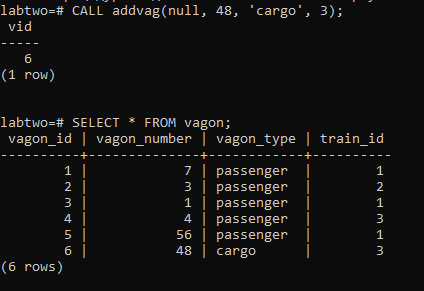


f. створення процедури з вхідним параметром та RETURN;

Створимо процедуру, що буде вставляти заданий рядок у таблицю vagon і поверне id запису:

CREATE OR REPLACE PROCEDURE addvag( VID INOUT INT, VN INT, VT VARCHAR(50), TI INT) LANGUAGE plpgsql AS $$ BEGIN INSERT INTO vagon(vagon\_number, vagon\_type, train\_id) VALUES (VN, VT, TI) RETURNING vagon\_id INTO VID; END $$;

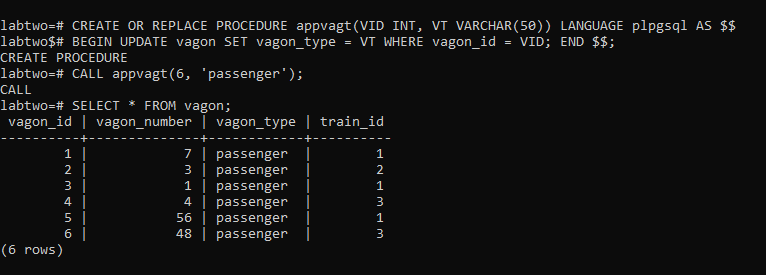




g. створення процедури оновлення даних в деякій таблиці БД;

Створимо процедуру, що оновить у таблиці vagon запис з вказаним id, а саме тип вагону:

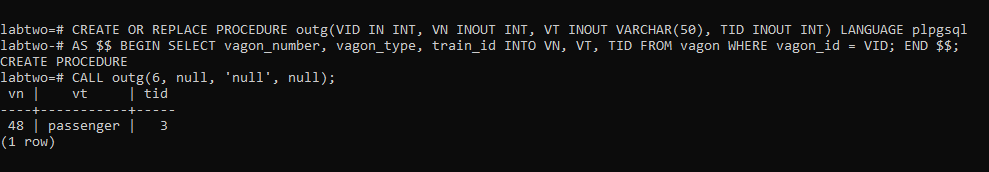
CREATE OR REPLACE PROCEDURE appvagt(VID INT, VT VARCHAR(50)) LANGUAGE plpgsql AS $$ BEGIN UPDATE vagon SET vagon\_type = VT WHERE vagon\_id = VID; END $$;



h. створення процедури, в котрій робиться вибірка даних.

Створимо процедуру, що повертає рядок з таблиці vagon за вказаним id:

CREATE OR REPLACE PROCEDURE outg(VID IN INT, VN INOUT INT, VT INOUT VARCHAR(50), TID INOUT INT) LANGUAGE plpgsql AS $$ BEGIN SELECT vagon\_number, vagon\_type, train\_id INTO VN, VT, TID FROM vagon WHERE vagon\_id = VID; END $$;

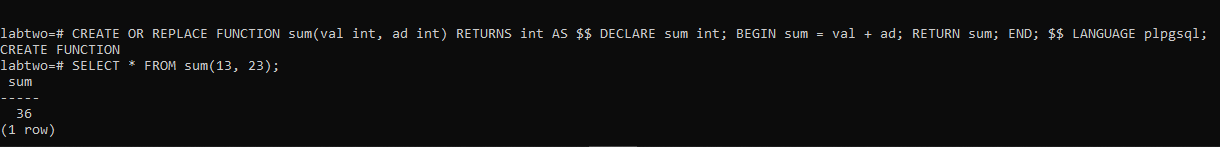


2) Функції:

a. створити функцію, котра повертає деяке скалярне значення;

Створимо функцію, що повертає суму двох вказаних цілих чисел:

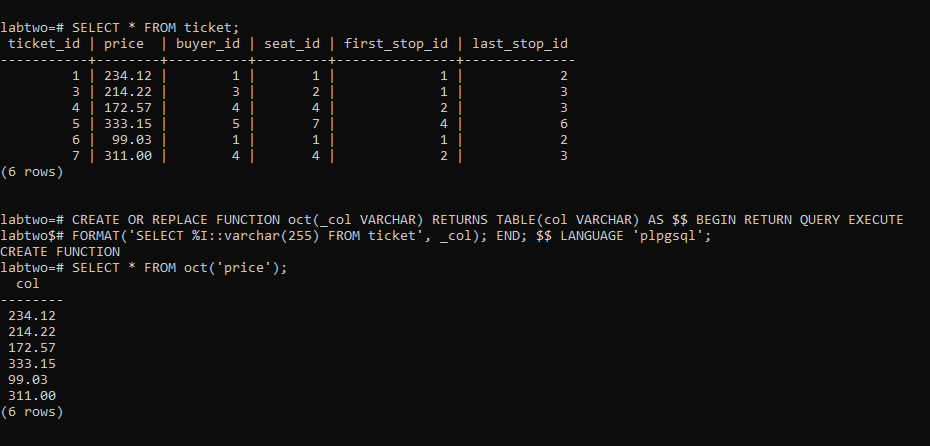
CREATE OR REPLACE FUNCTION sum(val int, ad int) RETURNS int AS $$ DECLARE sum int; BEGIN sum = val + ad; RETURN sum; END; $$ LANGUAGE plpgsql;



b. створити функцію, котра повертає таблицю з динамічним набором стовпців;

Створимо функцію, яка повертатиме вказаний за назвою стовпчик таблиці ticket:

CREATE OR REPLACE FUNCTION oct(\_col VARCHAR) RETURNS TABLE(col VARCHAR) AS $$ BEGIN RETURN QUERY EXECUTE FORMAT('SELECT %I::varchar(255) FROM ticket', \_col); END; $$ LANGUAGE 'plpgsql';

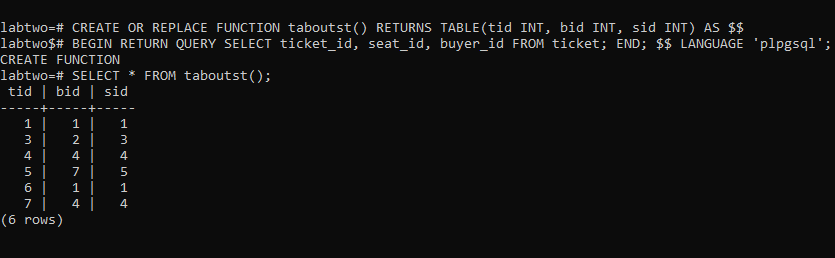


c. створити функцію, котра повертає таблицю заданої структури.

Створимо функцію, яка повертатиме ticket\_id, seat\_id, buyer\_id як нову таблицю з таблиці ticket:

CREATE OR REPLACE FUNCTION taboutst() RETURNS TABLE(tid INT, bid INT, sid INT) AS $$

BEGIN RETURN QUERY SELECT ticket\_id, seat\_id, buyer\_id FROM ticket; END; $$ LANGUAGE 'plpgsql';

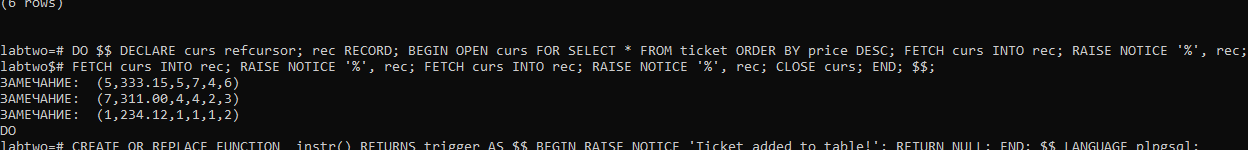


3) Робота з курсорами:

a. створити курсор, b. відкрити курсор, c. вибірка даних, робота з курсорами.

Створимо непідв’язаний курсор curs, відкриємо його для вибору записів з ticket, відсортованих за спаданням за price, виведемо перші три записи таблиці:

DO $$ DECLARE curs refcursor; rec RECORD; BEGIN OPEN curs FOR SELECT \* FROM ticket ORDER BY price DESC; FETCH curs INTO rec; RAISE NOTICE '%', rec; FETCH curs INTO rec; RAISE NOTICE '%', rec; FETCH curs INTO rec; RAISE NOTICE '%', rec; CLOSE curs; END; $$;



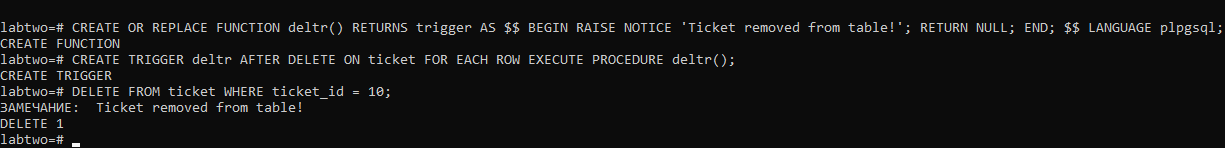
4) Робота з тригерами:

a. створити тригер, котрий буде спрацьовувати при видаленні даних;

Створимо тригер, котрий спрацьовуватиме при видалені рядка з таблиці ticket:

CREATE OR REPLACE FUNCTION deltr() RETURNS trigger AS $$ BEGIN RAISE NOTICE 'Ticket removed from table!'; RETURN NULL; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER deltr AFTER DELETE ON ticket FOR EACH ROW EXECUTE PROCEDURE deltr();

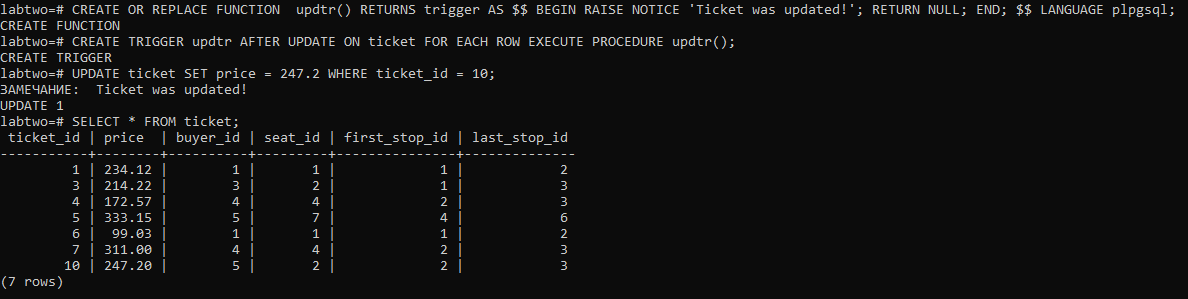


b. створити тригер, котрий буде спрацьовувати при модифікації даних;

Створимо тригер, котрий спрацьовуватиме при редагуванні рядка з таблиці ticket:

CREATE OR REPLACE FUNCTION updtr() RETURNS trigger AS $$ BEGIN RAISE NOTICE 'Ticket was updated!'; RETURN NULL; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER updtr AFTER UPDATE ON ticket FOR EACH ROW EXECUTE PROCEDURE updtr ();

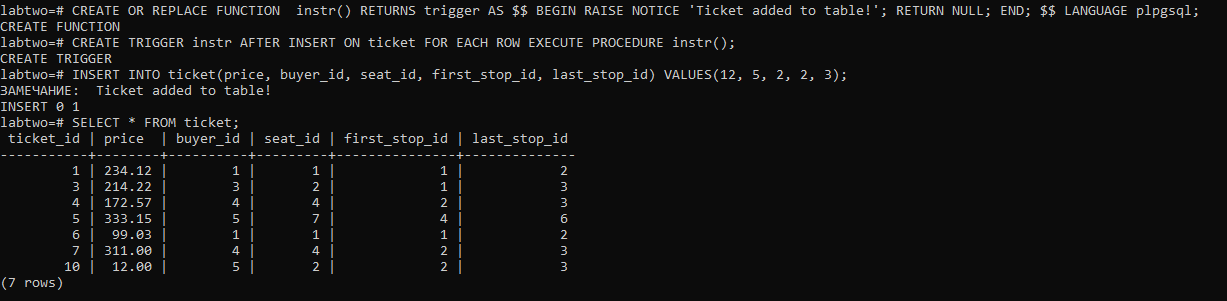


c. створити тригер, котрий буде спрацьовувати при додаванні даних.

Створимо тригер, котрий спрацьовуватиме при додаванні рядка з таблиці ticket:

CREATE OR REPLACE FUNCTION instr() RETURNS trigger AS $$ BEGIN RAISE NOTICE 'Ticket added to table!'; RETURN NULL; END; $$ LANGUAGE plpgsql;

CREATE TRIGGER instr AFTER INSERT ON ticket FOR EACH ROW EXECUTE PROCEDURE instr();



Висновок: під час виконання лабораторної роботи я навчився створювати складні запити, процедури, функції, курсори та тригери.