ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №7

Тема: «Користувацькі функції на мові запитів SQL»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-23

2025

Тема: «Користувацькі функції на мові запитів SQL».

Мета роботи: Ознайомлення з поняттям користувацьких функцій на мові запитів SQL, їх створенням та застосуванням.

Завдання лабораторної роботи:

1. Опрацювати теоретичний матеріал.

2. Відповідно до свого завдання написати не менше 3-х користувацьких функцій на мові запитів SQL.

Хід роботи

1. Опрацювання теоретичного матеріалу

Ознайомлено з теоретичним матеріалом щодо користувацьких функцій у PostgreSQL. Вивчено їх види (SQL-функції, PL/pgSQL, внутрішні, на мові C), синтаксис створення (CREATE FUNCTION), особливості виконання (розбір перед виконанням, заборона транзакційних команд). Розглянуто типи даних для аргументів (базові, складені), повернення значень (SETOF, TABLE, void), а також використання значень за замовчуванням.

2. Створення користувацьких функцій на мові запитів SQL

Для виконання завдання використано базу даних PartsTradeDB із таблицями Part і Mechanism. Створено три SQL-функції різних типів, які демонструють роботу з базовими типами, повернення таблиці та підмножини рядків.

Функція 1: Повернення загальної вартості запчастин для механізму

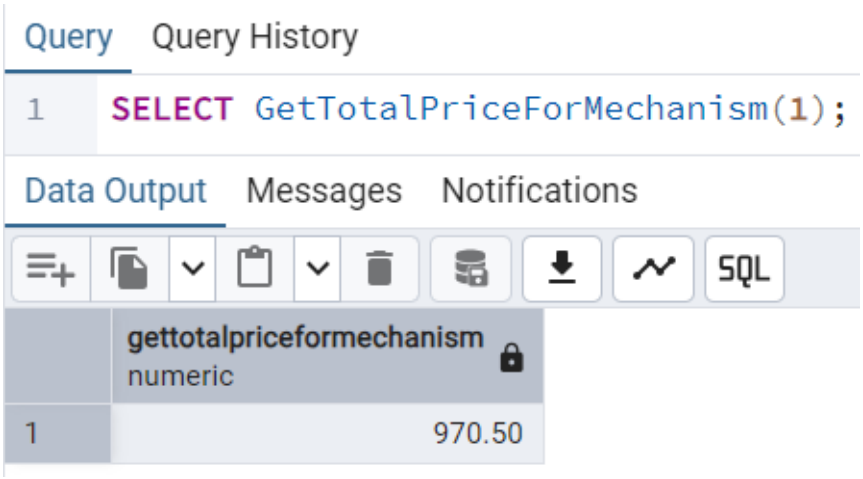
Функція приймає mech_id (базовий тип INTEGER), обчислює суму цін запчастин для заданого механізму і повертає значення типу DECIMAL. Використовує ANY для роботи з масивом Mechanism_IDs.

```sql
-- FUNCTION: public.gettotalpriceformechanism(integer)

-- DROP FUNCTION IF EXISTS public.gettotalpriceformechanism(integer);

CREATE OR REPLACE FUNCTION public.gettotalpriceformechanism(
    mech_id integer)
    RETURNS numeric
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
    SELECT SUM(p.Price)
    FROM Part p
    WHERE mech_id = ANY(p.Mechanism_IDs);
$BODY$;

ALTER FUNCTION public.gettotalpriceformechanism(integer)
    OWNER TO postgres;
```



Функція 2: Перелік запчастин для механізму

Функція повертає таблицю з трьома стовпцями (PartName, SerialNumber, Price). Використовує RETURNS TABLE для визначення структури результату. Об'єднує таблиці Part і Mechanism для вибірки.

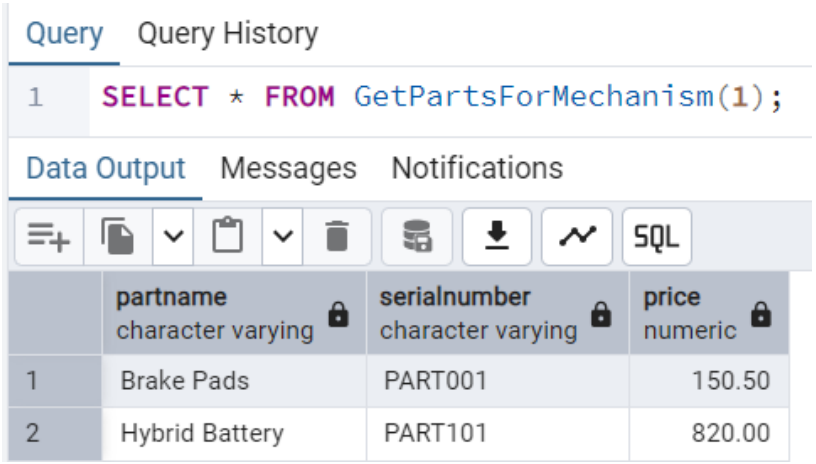
```
-- FUNCTION: public.getpartsformechanism(integer)

-- DROP FUNCTION IF EXISTS public.getpartsformechanism(integer);

CREATE OR REPLACE FUNCTION public.getpartsformechanism(
    mech_id integer)
    RETURNS TABLE(partname character varying, serialnumber character varying, price numeric)
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
    SELECT p.Name, p.Serial_Number, p.Price
    FROM Part p
    JOIN Mechanism m ON m.ID_Mechanism = ANY(p.Mechanism_IDs)
    WHERE m.ID_Mechanism = mech_id;
$BODY$;

ALTER FUNCTION public.getpartsformechanism(integer)
    OWNER TO postgres;
```



Функція 3: Підмножина запчастин із ціною вище заданої

Функція повертає підмножину рядків таблиці Part (тип SETOF Part), де ціна перевищує min_price. Використовує RETURNS SETOF для повернення набору рядків.

```
-- FUNCTION: public.getexpensiveparts(numeric)

-- DROP FUNCTION IF EXISTS public.getexpensiveparts(numeric);

CREATE OR REPLACE FUNCTION public.getexpensiveparts(
    min_price numeric)
    RETURNS SETOF part
    LANGUAGE 'sql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
    SELECT *
    FROM Part
    WHERE Price > min_price;
$BODY$;

ALTER FUNCTION public.getexpensiveparts(numeric)
    OWNER TO postgres;
```

Query | Query History

```
SELECT * FROM GetExpensiveParts(100);
```

Data Output | Messages | Notifications

Showing rows: 1 to 8 | Page No: 1 of 1

	id_part integer	name character varying (100)	serial_number character varying (50)	price numeric (10,2)	manufacturing_date date	stock_quantity integer	id_manufacturer integer	id_user integer	mechanism_ids integer[]
1	3	Shock Absorber	PART003	300.00	2024-10-20	20	3	3	{3}
2	1	Brake Pads	PART001	150.50	2024-12-01	50	1	[null]	{1}
3	4	Timing Belt	PART004	120.75	2024-09-10	40	4	[null]	{4}
4	365	Hybrid Battery	PART101	820.00	2024-07-10	15	2	1	{1}
5	366	Drum Brake Set	PART102	220.99	2024-06-15	35	3	2	{2}
6	367	Hydraulic Pump	PART103	410.75	2024-05-20	10	1	3	{3}
7	368	CVT Pulley	PART104	145.00	2024-04-12	25	4	4	{4}
8	369	Turbocharger	PART106	750.60	2024-02-28	8	2	1	{6}

Висновок: Лабораторна робота дозволила ознайомитися з користувацькими функціями на мові SQL у PostgreSQL на прикладі бази даних PartsTradeDB. Створено три функції: для обчислення суми цін (базовий тип), повернення таблиці запчастин (TABLE) та вибірки дорогих запчастин (SETOF). Функції успішно виконано, результати підтвердили їх коректність. Використання RETURNS TABLE і SETOF дозволило гнучко працювати з різними типами повернення даних.

Відповіді на контрольні питання

1. Назвіть види функцій представлених в PostgreSQL.
У PostgreSQL є чотири види функцій:

- Функції на мові запитів (SQL-функції).
- Функції на процедурних мовах (наприклад, PL/pgSQL, PL/Tcl, PL/Python).
- Внутрішні функції (вбудовані в ядро PostgreSQL).
- Функції на мові C (написані та скомпільовані на C).

2. Які типи даних можуть приймати функції в якості аргументів?
Функції в PostgreSQL можуть приймати:

- Базові типи (наприклад, INTEGER, DECIMAL, VARCHAR, BOOLEAN).
- Складені типи (наприклад, записи, масиви, композитні типи).
- Комбінації базових і складених типів (наприклад, масив цілих чисел INTEGER[]).
- Поліморфні типи (наприклад, ANY, ANYELEMENT для гнучкості).

3. Які оператори можуть використовуватися у функції SQL?
У SQL-функціях можна використовувати:

- Оператори для вибірки даних: SELECT.
- Оператори для модифікації даних: INSERT, UPDATE, DELETE (з RETURNING, якщо потрібно повернути результат).
- Інші SQL-команди (наприклад, WITH для CTE).
  Заборонено використовувати команди управління транзакціями (COMMIT, SAVEPOINT) та деякі допоміжні команди (VACUUM).

4. Які особливості синтаксису функції SQL?

- Функція створюється командою CREATE FUNCTION.
- Тіло функції записується як рядкова константа, зазвичай у $$ (знаки долара), щоб уникнути екранування апострофів.
- Вказується мова: LANGUAGE SQL.
- Тип повернення задається через RETURNS (наприклад, RETURNS DECIMAL, RETURNS TABLE, RETURNS SETOF).
- Останній оператор має відповідати типу повернення (наприклад, SELECT для повернення даних).
- Крапка з комою (;) розділяє оператори, але після останнього оператора її можна опустити.

5. Поясніть послідовність виконання функцій SQL.

- Розбір (Parsing): Спочатку PostgreSQL аналізує тіло функції, перевіряючи синтаксис і існування об’єктів (наприклад, таблиць). На цьому етапі команди, що змінюють системні каталоги (наприклад, CREATE TABLE), ще не впливають на подальший аналіз.
- Виконання (Execution): Після розбору виконуються оператори в порядку їх запису. Останній оператор (зазвичай SELECT або команда з RETURNING) визначає результат, який повертається.
- Якщо функція повертає SETOF або TABLE, повертаються всі рядки

результату останнього запиту.

6. Наведіть приклад функцій SQL з базовими типами та складними типами.

- **З базовим типом: Обчислення суми цін запчастин.**

```
CREATE FUNCTION GetTotalPrice() RETURNS DECIMAL AS $$
    SELECT SUM(Price)
    FROM Part;
$$ LANGUAGE SQL;
```

  - Повертає значення базового типу DECIMAL.
- **З складним типом: Повернення масиву цін (DECIMAL[]).**

```
CREATE FUNCTION GetPriceArray() RETURNS DECIMAL[] AS $$
    SELECT ARRAY_AGG(Price)
    FROM Part;
$$ LANGUAGE SQL;
```

  - Повертає складений тип DECIMAL[] (масив).

7. Наведіть приклад функції SQL з використанням значень за замовчуванням для вхідних і вихідних аргументів.

- **Функція фільтрує запчастини за мінімальною ціною (за замовчуванням 0).**

```
CREATE FUNCTION FilterPartsByPrice(min_price DECIMAL DEFAULT 0)
RETURNS TABLE (PartName VARCHAR, Price DECIMAL) AS $$
  SELECT Name, Price
  FROM Part
  WHERE Price >= min_price;
$$ LANGUAGE SQL;

SELECT * FROM FilterPartsByPrice();
SELECT * FROM FilterPartsByPrice(100);
```

8. Наведіть приклад функцій SQL, які повертають підмножину рядків таблиці.

- **Повернення запчастин із ціною вище 100 (тип SETOF).**

```
CREATE FUNCTION GetExpensiveParts()
RETURNS SETOF Part AS $$
  SELECT *
  FROM Part
  WHERE Price > 100;
$$ LANGUAGE SQL;

SELECT * FROM GetExpensiveParts();
```

9. Наведіть приклад функцій SQL, які повертають таблиці.

- **Повернення таблиці із запчастинами для заданого механізму.**

```
CREATE FUNCTION GetPartsForMechanism(mech_id INTEGER)
RETURNS TABLE (PartName VARCHAR, SerialNumber VARCHAR, Price
DECIMAL) AS $$
```

```
    SELECT p.Name, p.Serial_Number, p.Price
    FROM Part p
    JOIN Mechanism m ON m.ID_Mechanism = ANY(p.Mechanism_IDs)
    WHERE m.ID_Mechanism = mech_id;
$$ LANGUAGE SQL;

SELECT * FROM GetPartsForMechanism(1);
```

10. Наведіть приклад поліморфної функції SQL.

- **Функція, яка повертає перший елемент масиву будь-якого типу (ANYELEMENT).**

```
CREATE FUNCTION GetFirstElement(arr ANYARRAY)
RETURNS ANYELEMENT AS $$
    SELECT arr[1];
$$ LANGUAGE SQL;

SELECT GetFirstElement(ARRAY[1, 2, 3]::INTEGER[]);
SELECT GetFirstElement(ARRAY['a', 'b', 'c']::VARCHAR[]);
```