ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №8

Тема: «Віртуальні таблиці SQL»

Виконав:

Ст. *Лук'янчук Денис*

Група ПМІ-23

2025

Тема: «Віртуальні таблиці SQL».

Мета роботи: Ознайомлення з поняттям Віртуальні таблиці SQL, їх створення та застосування.

Завдання лабораторної роботи:

1. Опрацювати теоретичний матеріал.

2. Відповідно до свого завдання написати не менше 3-х представлень SQL. Виконання і результати проілюструвати знімками екрану, а також поясненнями їх роботи. Представлення мають бути різних типів – змінювані (з використанням CHECK OPTION), незмінювані, матеріалізовані.

Теоретичний матеріал

Представлення (Views) у SQL:

- Представлення — це віртуальна таблиця, створена на основі результату SQL-запиту. Вона не зберігає дані фізично (крім матеріалізованих представлень), а лише визначає, як отримувати дані з базових таблиць.
- Типи представлень:
    1. **Змінювані (Updatable Views)**:
        - Дозволяють виконувати операції INSERT, UPDATE, DELETE, якщо представлення відповідає певним умовам (наприклад, посилається на одну таблицю, не містить агрегатних функцій чи GROUP BY).
        - CHECK OPTION забезпечує, що зміни через представлення відповідають умовам WHERE у визначенні представлення.
    2. **Незмінювані (Non-Updatable Views)**:
        - Не дозволяють змінювати дані, якщо представлення включає складні запити (наприклад, з'єднання, агрегатні функції, GROUP BY).
        - Використовуються для спрощення доступу до даних або приховування складної логіки.
    3. **Матеріалізовані (Materialized Views)**:
        - Фізично зберігають дані, отримані з запиту, і можуть бути оновлені вручну або автоматично.
        - Використовуються для підвищення продуктивності при роботі з великими або складними наборами даних.

Переваги представлень:

- Спрощення запитів для користувачів.
- Обмеження доступу до даних (наприклад, показ лише певних стовпців).

- Забезпечення логічної абстракції над фізичною структурою бази даних.

Обмеження:

- Змінювані представлення мають обмеження на складність запитів.
- Матеріалізовані представлення потребують додаткового місця для зберігання та управління оновленнями.

Хід роботи

1. Змінюване представлення

Це представлення дозволить переглядати та змінювати дані про деталі (Part), які мають ціну менше 200. CHECK OPTION гарантуватиме, що будь-які вставки або оновлення через представлення відповідають цій умові.

```sql
-- View: public.affordableparts

-- DROP VIEW public.affordableparts;

CREATE OR REPLACE VIEW public.affordableparts
WITH (
  check_option=cascaded
) AS
 SELECT id_part,
    name,
    serial_number,
    price,
    stock_quantity,
    id_manufacturer
   FROM part
  WHERE price::numeric < 200::numeric;

ALTER TABLE public.affordableparts
    OWNER TO postgres;
```

Query | Query History

```sql
SELECT * FROM affordableparts
```

Data Output | Messages | Notifications

Showing rows: 1 to 12 | Page No: 1

	id_part integer	name character varying (100)	serial_number character varying (50)	price numeric (10,2)	stock_quantity integer	id_manufacturer integer
1	1	Brake Pads	PART001	150.50	50	1
2	2	Oil Filter	PART002	75.25	30	2
3	4	Timing Belt	PART004	120.75	40	4
4	5	Spark Plug	PART005	45.00	60	5
5	7	Fuel Pump	PART007	180.75	15	2
6	8	Air Filter	PART008	50.00	100	3
7	9	Wheel Bearing	PART009	90.50	30	4
8	14	Exhaust Pipe	PART014	110.00	40	4
9	15	Battery	PART015	150.00	50	5
10	16	Thermostat	PART016	45.25	60	1
11	17	Water Pump	PART017	95.50	35	2
12	18	Brake Caliper	PART018	175.00	18	3

Опис:

- **Тип:** Змінюване представлення.
- **Мета:** Показує деталі з таблиці Part, де ціна менша за 200. Дозволяє вставку та оновлення даних, але лише якщо Price < 200.
- **CHECK OPTION:** Забороняє вставку або оновлення записів, де Price >= 200.
- **Обмеження:** Представлення посилається лише на одну таблицю (Part) і не містить агрегатних функцій чи складних з'єднань, тому воно змінюване.

2. Незмінюване представлення

Це представлення покаже зведену інформацію про замовлення, включаючи ім’я користувача, кількість деталей і загальну суму замовлення. Через використання з'єднань і агрегатних функцій воно буде незмінюваним.

```sql
-- View: public.ordersummary

-- DROP VIEW public.ordersummary;

CREATE OR REPLACE VIEW public.ordersummary
 AS
 SELECT o.id_order,
    o.order_date,
    o.status,
    u.username,
    array_length(o.id_parts, 1) AS total_parts,
    sum(p.price::numeric * o.quantities[i.i]::numeric) AS total_amount
   FROM orders o
     JOIN appuser u ON o.id_user = u.id_user
     JOIN part p ON p.id_part = ANY (o.id_parts)
     JOIN LATERAL generate_subscripts(o.id_parts, 1) i(i) ON p.id_part = o.id_parts[i.i]
  GROUP BY o.id_order, o.order_date, o.status, u.username;

ALTER TABLE public.ordersummary
    OWNER TO postgres;
```

Query | Query History

```sql
SELECT * FROM ordersummary
```

Data Output | Messages | Notifications

Showing rows: 1 to 5 | Page No: 1

	id_order integer	order_date date	status character varying (50)	username character varying (100)	total_parts integer	total_amount numeric
1	4	2025-03-04	New	Anna	1	120.75
2	5	2025-03-05	Processing	Pavlo	1	90.00
3	2	2025-03-02	Processing	Maria	1	75.25
4	1	2025-03-01	New	Oleg	1	301.00
5	3	2025-03-03	Delivered	Igor	1	900.00

Опис:

- **Тип:** Незмінюване представлення.
- **Мета:** Показує деталі замовлень, включаючи ім’я користувача, кількість деталей у замовленні та загальну суму (ціна деталі × кількість).

- **Чому незмінюване:** Використовує з'єднання (JOIN), агрегатну функцію (SUM), GROUP BY і розгортання масивів (generate_subscripts), що робить його непридатним для операцій INSERT/UPDATE.
- **Особливості:** Використовує generate_subscripts для коректного зіставлення цін деталей з кількостями в масивах ID_Parts і Quantities.

3. Матеріалізоване представлення

Це представлення зберігатиме статистику про кількість деталей для кожного виробника та механізму, що корисно для аналізу запасів. Дані фізично зберігаються і можуть бути оновлені за потреби.

```sql
-- View: public.manufacturermechanismstats

-- DROP MATERIALIZED VIEW IF EXISTS public.manufacturermechanismstats;

CREATE MATERIALIZED VIEW IF NOT EXISTS public.manufacturermechanismstats
TABLESPACE pg_default
AS
 SELECT m.name AS manufacturer_name,
    mech.name AS mechanism_name,
    count(p.id_part) AS part_count,
    sum(p.stock_quantity::integer) AS total_stock
   FROM manufacturer m
     LEFT JOIN part p ON p.id_manufacturer = m.id_manufacturer
     LEFT JOIN mechanism mech ON mech.id_mechanism = ANY (p.mechanism_ids)
  GROUP BY m.name, mech.name
WITH DATA;

ALTER TABLE IF EXISTS public.manufacturermechanismstats
    OWNER TO postgres;
```

```sql
SELECT * FROM manufacturermechanismstats
```

Data Output | Messages | Notifications

Showing rows: 1

	manufacturer_name character varying (100)	mechanism_name character varying (100)	part_count bigint	total_stock bigint
1	Skoda	Steering	2	65
2	Bosch	Brake System	1	25
3	Ford	Brake System	1	18
4	Ford	Engine	2	120
5	Renault	Transmission	1	40
6	Bosch	Transmission	1	8
7	Renault	Engine	1	40
8	Bosch	Engine	2	110
9	Skoda	Engine	2	60
10	Renault	Suspension	2	52
11	Toyota	Engine	3	62
12	Toyota	Brake System	1	30
13	Ford	Suspension	1	20

Опис:

- **Тип:** Матеріалізоване представлення.
- **Мета:** Зберігає статистику про кількість деталей і загальний запас (Stock_Quantity) для кожного виробника та механізму.
- **Особливості:**
    - Дані зберігаються фізично, що зменшує час виконання запитів порівняно з віртуальним представленням.
    - Оновлення виконується за допомогою REFRESH MATERIALIZED VIEW ManufacturerMechanismStats.
    - Використовує LEFT JOIN, щоб включити всіх виробників, навіть якщо у них немає деталей.
- **Чому матеріалізоване:** Дозволяє кешувати результати складного запиту для швидкого доступу, особливо якщо дані рідко змінюються.

Висновок:

На основі наданої схеми бази даних створено три SQL-представлення: змінюване з `CHECK OPTION` (`AffordableParts`), незмінюване (`OrderSummary`) та матеріалізоване (`ManufacturerMechanismStats`). Змінюване представлення дозволяє оновлювати деталі з ціною до 200, незмінюване агрегує дані замовлень, а матеріалізоване зберігає статистику запасів для швидкого доступу. Кожне представлення супроводжується SQL-кодом, прикладами запитів та інструкціями для відтворення. Робота демонструє різні типи представлень, їх практичне застосування та відповідність структурі бази даних, забезпечуючи зручність аналізу та управління даними.

Відповіді на контрольні питання

1. Що означає поняття Представлення (VIEWS) та для чого вони застосовуються?

Визначення: Представлення — це віртуальна таблиця, створена на основі SQL-запиту, яка відображає дані з базових таблиць без фізичного зберігання (крім матеріалізованих представлень).
Застосування:

- Спрощують складні запити для користувачів.
- Обмежують доступ до даних, показуючи лише потрібні стовпці чи рядки.
- Забезпечують абстракцію, дозволяючи змінювати структуру бази без впливу на запити.
- Використовуються для звітів, аналітики та підвищення безпеки.

2. Які умови має задовольняти змінюване представлення?

Змінюване представлення дозволяє `INSERT`, `UPDATE`, `DELETE` і має відповідати умовам:

- Посилається на одну базову таблицю.
- Не містить `JOIN`, агрегатних функцій (`SUM`, `COUNT`), `GROUP BY`, `DISTINCT`, підзапитів чи `UNION`.
- Стовпці відповідають базовій таблиці без обчислень.
- Усі `NOT NULL` стовпці без значень за замовчуванням включені.
- Відсутні спеціальні тригери чи правила, що перешкоджають змінюваності.

3. **Поясніть зміну змінюваного представлення, якщо представлення це віртуальна таблиця, то що ж змінюється?**

Змінюване представлення діє як інтерфейс до базової таблиці. Операції `INSERT`, `UPDATE`, `DELETE` через представлення змінюють дані в базовій таблиці, а не в самому представленні. PostgreSQL автоматично транслює ці операції, забезпечуючи оновлення базової таблиці, а представлення відображає актуальний результат запиту.

4. Поясніть для чого в представленні вказується CHECK OPTION.

`CHECK OPTION` у змінюваних представленнях гарантує, що `INSERT` або `UPDATE` відповідають умові `WHERE` представлення. Це запобігає додаванню чи оновленню даних, які не відповідають фільтру, забезпечуючи цілісність даних у контексті представлення. Наприклад, якщо представлення показує записи з `price < 100`, `CHECK OPTION` блокує вставку записів із `price >= 100`.

5. Що таке матеріалізоване представлення?

Матеріалізоване представлення фізично зберігає результати SQL-запиту на диску, діючи як кешована таблиця. На відміну від віртуальних представлень, воно не оновлюється автоматично і потребує команди `REFRESH MATERIALIZED VIEW` для синхронізації. Використовується для оптимізації складних запитів і аналітики, коли швидкість доступу важливіша за актуальність.

6. Назвіть переваги представлень.

- Спрощують запити, приховуючи складну логіку.
- Підвищують безпеку, обмежуючи доступ до даних.
- Забезпечують абстракцію над структурою бази.
- Сприяють зручному аналізу та звітності.
- Матеріалізовані представлення покращують продуктивність завдяки кешуванню.

7. Які недоліки властиві представленням?

- Більшість представлень незмінювані через складні запити.
- Звичайні представлення можуть бути повільними для складних обчислень.
- Матеріалізовані представлення потребують оновлення та додаткового місця.
- Зміни в базових таблицях можуть порушити роботу представлень.
- Складність відлагодження при помилках у запитах.
- Обмеження на тригери та деякі операції.