Контрольна Робота з Бази даних та інформаційних систем

Захаров Дмитро

14 листопада, 2024

Варіант 5

Зміст

1 Завдання 1: Мережва Модель Даних

Умова Задачі 1.1. Мережева модель даних.

Відповідь. Мережева модель даних була дуже поширеною на певному етапі розвитку баз даних і досі використовується в деяких системах. Вона подібна до ієрархічної моделі, оскільки також складається з набору записів, які можуть бути власниками або членами групи, але головна відмінність полягає в тому, що один запис у мережевій моделі може бути частиною кількох груп одночасно. Кожна група має свою назву, а типи групових відносин відрізняються від їхніх конкретних екземплярів. Тип визначає спільні характеристики для всіх екземплярів, а сам екземпляр є записом-власником із підлеглими елементами. При цьому обмеження таке: один запис не може одночасно бути частиною двох груп одного типу.

У цій моделі будь-які елементи можуть бути пов'язані один з одним, а підлеглий запис може мати більше одного власника. Це дозволяє підтримувати зв'язки типу "багато-до-багатьох". Щоб спростити мережеву структуру, можна додати надмірні дані, що дозволяє створити дерево через дублювання деяких елементів.

Іноді така надмірність є прийнятною, але в деяких випадках вона може бути значною. Методи, що добре працюють для деревовидних структур, часто не підходять для мережевих моделей, через що програми для обробки дерев не можуть працювати з мережею.

Ситуацію ускладнює те, що методи, які застосовуються до одного виду мережевої структури, можуть бути неефективними для іншого виду. Суттєвим мінусом мережевої моделі є необхідність суворого дотримання фізичної структури даних, що ускладнює організацію запитів. Однак її ключова перевага – можливість створювати складні моделі, які краще відповідають реальним процесам. Щоб зменшити складність ієрархічної та мережевої моделей, було розроблено реляційну модель.

Сильні сторони мережевої моделі — це її ефективність у використанні пам'яті та швидкість обробки. У порівнянні з ієрархічною, мережева модель точніше відображає предметну область завдяки додатковим зв'язкам між елементами. Проте серед її мінусів — складність у використанні через жорстку схему. Контроль цілісності даних у мережевій моделі послаблений через можливість довільних зв'язків, і при зміні даних часто потрібно змінювати програмне забезпечення.

2 Завдання 2: Псевдоніми

Умова Задачі 2.1. Псевдоніми в SQL

Відповідь. У SQL псевдоніми (aliases) використовуються для тимчасового перейменування таблиць або стовпців у запитах. Це дозволяє зробити код більш зрозумілим і зручним для читання, особливо при роботі зі складними виразами чи кількома таблицями.

Псевдонім для стовпця дозволяє змінити назву виведеного результату на більш зрозумілу для користувача. Для цього використовується ключове слово `AS`, хоча його можна опустити. Наприклад,

```
SELECT
    AVG(math_score) AS avg_math_score,
    AVG(age) AS avg_age
FROM students;
```

вибере середній бал з математики та вік учнів з таблиці `students` та перейменує їх на `avg_math_score` та `avg_age` відповідно.

Альтернативно, псевдоніми можна використовувати для усієї таблиці. Наприклад, якщо ми хочемо взяти бали з математики та вік з таблиці `students`, то можна використати псевдоніми для усієї таблиці:

```
SELECT s.math_score, s.age
FROM students s;
```

Для наступної таблиці, ця команда цілком працює:

```
-- Create the tables
CREATE TABLE students (
    student_id INT PRIMARY KEY,
        name       VARCHAR(50),
    math_score INT,
    age        INT
);

-- Insert data into students table
INSERT INTO students (student_id, name, math_score, age) VALUES
(1, 'Dmytro', 95, 15),
(2, 'Olena', 100, 14),
(3, 'Mykhailo', 70, 15),
(4, 'John', 40, 14);
```

Помітимо, що тут `AS` є необов'язковим, але він може полегшити читання запиту.

Окрім цього, псевдоніми можна створювати з декількох слів через пробіл. Наприклад,

```
SELECT name AS [Super Guys]
FROM students;
```

Альтернативно, можна було використати

```
SELECT name AS "Super Guys"
FROM students;
```

Ба більше, в якості псевдоніму можна конкатенувати рядки:

```
SELECT
    student_id as "Student ID",
    CONCAT(name, ', ', math_score, ', ', age) AS "Student Info"
FROM students;
```

Результатом буде таблиця:

```
Student ID  Student Info
1           Dmytro, 95, 15
2           Olena, 100, 14
3           Mykhailo, 70, 15
4           John, 40, 14
```

Замітка. Команда `CONCAT` використовується для конкатенації рядків, проте, наскільки мені відомо, вона підтримується не усюди (наприклад, підтримується у `MySQL`).

3 Завдання 3: Середні значення

Умова Задачі 3.1. При вибірці з таблиці employees створіть нові псевдоніми `avg_sal`, `avg_age` в яких будуть лежати середні арифметичні зарплати (`salary`) і віку (`age`). Підрахуйте для усієї таблиці.

Відповідь. Для початку, давайте вкажемо міграцію для створення таблиці та заповнення її тестовими даними:

```sql
-- Create the employees table
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    name VARCHAR(50),
    salary DECIMAL(10, 2),
    age INT
);

-- Insert data into employees table
INSERT INTO employees (employee_id, name, salary, age) VALUES
(1, 'John Doe', 50000.00, 30),
(2, 'Jane Smith', 60000.00, 40),
(3, 'Alice Johnson', 55000.00, 35),
(4, 'Bob Brown', 48000.00, 28);
```

Тепер, можемо використати псевдоніми для підрахунку середніх значень зарплати та віку:

```sql
-- Query to calculate average salary and age
-- Query to select average salary and age with aliases
SELECT
    AVG(salary) AS avg_sal,
    AVG(age) AS avg_age
FROM employees;
```

Виходом буде:

```
      avg_sal    | avg_age
   53250.000000 | 33.2500
```

4 Завдання 4: Вкладені запити

Умова Задачі 4.1. Задані таблиці

$$\begin{aligned} \texttt{table1}: &\quad \texttt{user\_id}, \texttt{username}, \texttt{dpt\_name} \\ \texttt{table2}: &\quad \texttt{phone\_id}, \texttt{dpt\_name}, \texttt{phone\_number} \\ \texttt{table3}: &\quad \texttt{room\_id}, \texttt{phone\_number}, \texttt{room\_name} \end{aligned}$$

Необхідно вибрати номери кімнат в яких сидить користувачі з ім'ям `Nick` Не вживати `JOIN`.

Відповідь. Для початку, давайте вкажемо міграцію для створення таблиць та заповнення їх даними:

```sql
-- Create the tables
CREATE TABLE table1 (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    dpt_name VARCHAR(50)
);

CREATE TABLE table2 (
    phone_id INT PRIMARY KEY,
    dpt_name VARCHAR(50),
    phone_number VARCHAR(20)
);

CREATE TABLE table3 (
    room_id INT PRIMARY KEY,
    phone_number VARCHAR(20),
    room_name VARCHAR(50)
);

-- Insert data into table1
INSERT INTO table1 (user_id, username, dpt_name) VALUES
(1, 'Nick', 'HR'),
(2, 'Alice', 'IT'),
(3, 'Nick', 'Finance'),
(4, 'John', 'HR');

-- Insert data into table2
INSERT INTO table2 (phone_id, dpt_name, phone_number) VALUES
(1, 'HR', '12345'),
(2, 'IT', '67890'),
(3, 'Finance', '11223'),
(4, 'Marketing', '44556');

-- Insert data into table3
```

```
INSERT INTO table3 (room_id, phone_number, room_name) VALUES
(1, '12345', 'Room A'),
(2, '67890', 'Room B'),
(3, '11223', 'Room C'),
(4, '44556', 'Room D');
```

Далі, скористаємось функцією вкладених запитів для вирішення поставленої задачі:

```
-- Query to select room names where users named 'Nick' sit
SELECT room_name
FROM table3
WHERE phone_number IN (
    SELECT phone_number
    FROM table2
    WHERE dpt_name IN (
        SELECT dpt_name
        FROM table1
        WHERE username = 'Nick'
    )
);
```

Виходом буде:

```
   room_name
     Room A
     Room C
```

Тут, ми спочатку вибираємо назви відділів, в яких працює користувач з ім'ям `Nick`. Потім, вибираємо номери телефонів, які належать цим відділам. Нарешті, вибираємо назви кімнат, в яких знаходяться телефони з вибраними номерами.

Відмітка. Ця команда працює лише за умови, що телефон з таблиці 2 відповідає саме цілому відділу, а не окремому користувачеві, бо інакше гарантувати зв'язок між користувачем та його телефоном буде неможливо, які б команди ми не використовували.