Rakamin Academy

id/x partners

Project Based Internship

SQL Operations

Basic Table Operation

Set Operators

Daftar Isi

Exploration Source

[Article]

- SQL CREATE TABLE, ALTER TABLE, DROP TABLE, TRUNCATE TABLE
- MySQL Subquery
- SQL Introduction

[Video]

- All Types of SQL Commands with Syntax & Example
- SQL Commands With Interview Questions
- Mengenal Istilah DDL, DML, DCL pada Basis Data SQL
- Learn Basic SQL in 15 Minutes | Business Intelligence For Beginners |
- SQL Tutorial For Beginners 1/3

A. *Introduction*

SQL (Structured Query Language) adalah bahasa pemrograman yang cukup *powerful* untuk mengelola dan memanipulasi data dalam *relational database*. Ini menyediakan berbagai operasi untuk memperoleh, memperbarui, dan menghapus data. Pada artikel ini, kita akan mengeksplorasi tiga operasi SQL penting: operasi tabel dasar, *operator set*, dan ekspresi tabel umum, dengan contoh penerapannya dan studi kasus.

B. *Basic Table Operation*

SQL menyediakan sekumpulan operasi dasar yang memungkinkan kita mengelola data dalam sebuah tabel. Klausa SQL yang biasa digunakan digolongkan kedalam beberapa operasi yang didefinisikan sebagai *data manipulation language* (DML), *data definition language* (DDL), *data query language* (DQL), dan *data control language* (DCL).

- *Data Definition Language* (DDL) berisi kumpulan klausa SQL yang memiliki fungsi untuk melakukan pembentukan definisi dari sebuah tabel. Beberapa klausa yang termasuk dalam DDL antara lain adalah *CREATE, ALTER, DROP, TRUNCATE*.
- *Data Manipulation Language* (DML) berisi kumpulan klausa SQL yang memiliki fungsi untuk melakukan manipulasi pemilihan data dalam sebuah tabel ataupun *view*. Beberapa klausa yang termasuk dalam DML ini adalah *INSERT, UPDATE, DELETE, MERGE*.
- *Data Query Language* (DQL) berisi klausa SELECT yang digunakan untuk memilih data dalam sebuah tabel dan view.

- *Data Control Language* (DCL) berisikan klausa SQL untuk mengontrol kendali pengguna atas sebuah tabel dalam database. Beberapa klausa yang termasuk dalam DCL antara lain *GRANT* dan *REVOKE*.

Berikut ini merupakan definisi dan contoh penggunaan dari beberapa klausa SQL yang umum digunakan :

- CREATE TABLE

  CREATE TABLE merupakan query yang digunakan untuk membuat tabel baru, berikut merupakan contoh querynya:

```
CREATE TABLE products (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  price DECIMAL(10, 2) NOT NULL
);
```

  Untuk membuat tabel baru dalam suatu database querynya adalah dengan memanggil *statement CREATE TABLE* diikuti dengan nama tabelnya dan kolom yang akan ada pada tabel tersebut diikuti dengan jenis datanya.

- *SELECT*

  Pernyataan *SELECT* digunakan untuk mengambil data dari satu atau lebih tabel. Berikut adalah contoh pernyataan *SELECT*:

```
SELECT id, name, price
FROM products
WHERE price > 10 OR id > 5
ORDER BY price DESC
LIMIT 3 ;
```

Query ini mengambil kolom id, name, dan *price* dari tabel produk di mana harganya lebih besar dari 10 atau id lebih besar dari 5 dan mengurutkan hasilnya berdasarkan harga dalam urutan menurun, kemudian membatasi *output* hanya 3 baris pertama.

Seringkali kita juga mendengar istilah Select Distinct pada perintah SQL. Lalu apa perbedaan Select dan Select Distinct? Select akan mengambil seluruh *value* pada kolom yang dipilih, sedangkan Select Distinct akan mengeluarkan hasil *value* yang unik/tidak *duplicate* satu sama lain. Contoh :

Table Products

id	name	price
1	Buku	9
2	Kertas	5
3	Buku	8
4	Buku	5

Query :

```
SELECT DISTINCT NAME
FROM products
```

Hasil :

name
Buku
Kertas

Dapat dilihat pada tabel products terdapat 3 baris Buku dan 1 baris Kertas. Hasil dari perintah Select Distinct untuk kondisi tabel products adalah 1 baris Buku dan 1 baris Kertas. Select Distinct hanya akan menampilkan *value* unik dan tidak *duplicate*. Seperti itulah gambaran sederhana output dari Select Distinct pada SQL.

Berikut adalah beberapa opsi lain yang dapat digunakan ketika mengolah data dalam perintah Select.

1. Operasi *Grouping (Group By, Having, Rollup)*

   A. *Group By*: Mengelompokkan data berdasarkan kolom tertentu

```
SELECT department, COUNT(*) AS employee_count
FROM employees
GROUP BY department;
```

   B. *Having*: Menerapkan kondisi setelah operasi *Group By*

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

   C. *Rollup*: Membuat subtotal dan total secara hierarkis

```
SELECT department, city, COUNT(*) AS employee_count
FROM employees
GROUP BY ROLLUP(department, city);
```

Penjelasan :

- Operasi *GROUP BY* digunakan untuk mengelompokkan baris berdasarkan nilai tertentu (contohnya, departemen).
- Operasi *HAVING* digunakan untuk memberlakukan kondisi pada hasil setelah operasi *GROUP BY* (contohnya, rata-rata gaji di atas 50.000).
- Operasi *ROLLUP* digunakan untuk membuat subtotal dan total hierarkis berdasarkan kolom yang ditentukan.

2. Operasi *Basic Joining* (*Inner, Outer, Left, Right, Full Outer*) dan *Aliasing*

A. *Inner Join*: Mengambil baris yang memiliki nilai yang cocok di kedua tabel

```
SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

B. *Left Join*: Mengambil semua baris dari tabel kiri dan yang cocok dari tabel kanan

```
SELECT customers.customer_name, COUNT(orders.order_id) AS
order_count
FROM customers
LEFT JOIN orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_name;
```

C. *Right Join*: Mengambil semua baris dari tabel kanan dan yang cocok dari tabel kiri

```
SELECT customers.customer_name, orders.order_id
FROM customers
RIGHT JOIN orders ON customers.customer_id = orders.customer_id;
```

D. *Full Outer Join*: Mengambil semua baris dari kedua tabel, mencocokkan di mana mungkin

```
SELECT customers.customer_name, orders.order_id
FROM customers
FULL OUTER JOIN orders ON customers.customer_id = orders.customer_id;
```

E. *Alias*: Memberikan alias pada tabel untuk memudahkan penulisan query

```
SELECT c.customer_name, o.order_id
FROM customers AS c
JOIN orders AS o ON c.customer_id = o.customer_id;
```

Penjelasan:

- *INNER JOIN* menggabungkan baris dari kedua tabel berdasarkan nilai yang cocok.
- *LEFT JOIN* mengambil semua baris dari tabel kiri dan yang cocok dari tabel kanan.
- *RIGHT JOIN* mengambil semua baris dari tabel kanan dan yang cocok dari tabel kiri. Dalam contoh ini, hasilnya akan mencakup semua pesanan (*orders*) dan data pelanggan (*customers*) yang memiliki nilai yang cocok pada kolom customer_id. Jika tidak ada nilai yang cocok dari tabel kiri, kolom-kolom dari tabel kiri akan berisi nilai NULL dalam hasil query.
- *FULL OUTER JOIN* mengambil semua baris dari kedua tabel, mencocokkan di mana mungkin.
- *AS* digunakan untuk memberikan alias pada tabel agar lebih mudah ditulis dalam query.

- *INSERT*

  Pernyataan *INSERT* digunakan untuk menambahkan data baru ke tabel. Berikut adalah contoh pernyataan *INSERT*:

```
INSERT INTO products (id, name, price)
VALUES (1, 'Product A', 20.99);
```

Pernyataan ini menambahkan data baru dengan id 1, name "Produk A", dan price 20,99 ke tabel product.

- *UPDATE*

  Pernyataan *UPDATE* digunakan untuk mengubah data yang ada. Berikut adalah contoh pernyataan *UPDATE*:

```
UPDATE products
SET price = 19.99
WHERE id = 1;
```

  Query ini mengubah harga yang ada dengan id 1 menjadi 19,99.

- *DELETE*

  Pernyataan *DELETE* digunakan untuk menghapus data dari tabel. Berikut adalah contoh pernyataan *DELETE*:

```
DELETE FROM products
WHERE id = 1;
```

  Pernyataan ini menghapus data dengan id 1 dari tabel produk.

C. Set Operator

SQL juga menyediakan set operator yang memungkinkan kita menggabungkan hasil dari dua atau lebih pernyataan SELECT. Operator ini termasuk *UNION, INTERSECT,* dan *EXCEPT*. Operator *UNION* digunakan untuk menggabungkan hasil dari dua atau lebih pernyataan *SELECT* menjadi satu set hasil. Operator

INTERSECT mengembalikan baris umum antara dua pernyataan *SELECT*, sedangkan operator *EXCEPT* mengembalikan baris yang ada di pernyataan *SELECT* pertama tetapi tidak di pernyataan *SELECT* kedua. Pada kali ini kita akan membuat dua buah tabel yang akan digunakan pada set operator.

```
CREATE TABLE sales (
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    sale_date DATE NOT NULL
);

CREATE TABLE inventory (
    product_id INT NOT NULL,
    quantity INT NOT NULL
);
```

Tabel yang terbentuk adalah tabel *sales* dan *inventory*, kedua tabel ini akan digunakan sebagai contoh *SET OPERATOR* pada sub-topik berikutnya.

- *UNION*

  Operator *UNION* digunakan untuk menggabungkan hasil dari dua atau lebih pernyataan *SELECT* menjadi satu set hasil. Perlu diperhatikan bahwa ketika kita menggunakan *UNION*, maka posisi kolom yang digabungkan harus memiliki tipe data dan struktur yang sama. Berikut adalah contoh penggunaan operator UNION:

```
SELECT product_id, quantity, sale_date
FROM sales
UNION
```

```
SELECT product_id, quantity, NULL
FROM inventory;
```

Pernyataan ini menggabungkan hasil dari dua pernyataan SELECT: yang pertama memilih product_id, quantity, dan sale_date dari tabel sales, dan yang kedua memilih product_id, quantity, dan NULL dari tabel inventori.

- INTERSECT

  Operator INTERSECT mengembalikan data umum antara dua pernyataan SELECT. Berikut adalah contoh operator INTERSECT:

```
SELECT product_id
FROM sales
INTERSECT
SELECT product_id
FROM inventory;
```

  Pernyataan ini mengembalikan product_id yang ada di tabel sales dan inventory.

- EXCEPT

  Operator EXCEPT mengembalikan baris yang ada di pernyataan SELECT pertama tetapi tidak di pernyataan SELECT kedua. Berikut adalah contoh operator EXCEPT:

```
SELECT product_id
FROM sales
EXCEPT
SELECT product_id
FROM inventory;
```

Query ini hanya akan mengembalikan baris yang muncul di tabel sales tetapi tidak di tabel inventory

CASE STUDY : FINANCIAL TECHNOLOGY PLATFORM

Misalkan Anda bekerja untuk *client* perusahaan *financial technology (fintech)* yang menyediakan *platform* untuk layanan keuangan, seperti pembayaran digital, pinjaman, dan investasi. Perusahaan Anda memiliki sistem informasi yang mengumpulkan data transaksi, informasi pengguna, dan aktivitas keuangan dari pelanggan.

Salah satu tugas Anda adalah melakukan analisis data untuk mengidentifikasi tren dan pola yang dapat meningkatkan pengalaman pengguna, mengurangi risiko, dan meningkatkan efisiensi operasional.

SOLUSI

```
WITH user_transactions AS (
SELECT user_id, SUM(amount) AS total_spending, COUNT(*) AS transaction_count
 FROM transactions
  GROUP BY user_id
),
loan_performance AS (
SELECT user_id, AVG(loan_amount) AS average_loan_amount, MAX(loan_status) AS
highest_loan_status
 FROM loans
 GROUP BY user_id
```

```
)
SELECT u.country, COUNT(DISTINCT u.user_id) AS users,
    SUM(ut.total_spending) AS total_spending,
    SUM(ut.transaction_count) AS total_transactions,
    AVG(lp.average_loan_amount) AS avg_loan_amount,
    MAX(lp.highest_loan_status) AS highest_loan_status
FROM users u
JOIN user_transactions ut ON u.user_id = ut.user_id
LEFT JOIN loan_performance lp ON u.user_id = lp.user_id
GROUP BY u.country;
```

Dalam query ini, dua CTE ditentukan: "**user_transactions**" menghitung total pengeluaran dan jumlah transaksi untuk setiap pengguna, dan "**loan_performance**" menghitung rata-rata jumlah pinjaman dan status pinjaman tertinggi untuk setiap pengguna yang telah mengajukan pinjaman. Pernyataan SELECT utama kemudian menggabungkan CTE dengan tabel pengguna, transaksi, dan pinjaman untuk memberikan gambaran holistik tentang aktivitas keuangan pengguna dalam suatu negara.

REFERENCE

Table operations in MS SQL Server - GeeksforGeeks

10 SQL Operations for 80% of your Data Manipulation | by Anmol Tomar

DBMS SQL Set Operation - javatpoint

SQL Server Common Table Expressions (CTE)

Mengenal DDL, DML dan DCL di MySQL - ePlusGo