MODUL 4 PENGGUNAAN SQL

MODUL 4 Penggunaan SQL

Tujuan :

Setelah menyelesaikan modul ini, mahasiswa diharapkan dapat :

1. Mengetahui perbedaan query DDL dan DML.
2. Mengetahui dan menggunakan klausa dan operator yang ada pada SQL.

Perintah yang digunakan *Data Definition Language* (DDL)

a. CREATE
   - CREATE SCHEMA / CREATE DATABASE
   - CREATE TABLE
   - CREATE VIEW

b. ALTER
   - ALTER SCHEMA / ALTER DATABASE
   - ALTER TABLE
   - ALTER VIEW

c. DROP
   - DROP SCHEMA
   - DROP TABLE
   - DROP VIEW

d. CONSTRAINT, contoh:
   - CREATE TABLE … FOREIGN KEY … REFERENCES…
   - ALTER TABLE … ADD FOREIGN KEY … REFERENCES …
   - ALTER TABLE … DROP FOREIGN KEY …

Key Pada Implementasi Tabel

Setiap *Key* yang diciptakan pada tabel memiliki fungsi tersendiri :

1. **FOREIGN KEY ➔** Konsep tabel induk dan tabel anak. Merujuk pada tabel lain yang memiliki PRIMARY KEY, untuk menjaga *integrity constraint*.
2. **INDEX KEY ➔** Untuk mempercepat pencarian. Efek samping: memperlambat proses INSERT, UPDATE dan DELETE.
3. **UNIQUE KEY ➔** Nilai tiap-tiap record atau tupel hanya boleh ada satu.
4. **PRIMARY KEY ➔** Berfungsi ganda, yakni sebagai referensi dari FOREIGN KEY, sebagai INDEX KEY dan juga sebagai UNIQUE KEY.

Integrity Constraint

Merupakan batasan-batasan yang diberikan pada skema basis data, tujuannya untuk menjaga konsistensi data.

Klausa-klausa yang termasuk dalam ***constraint*** adalah:

1. **NOT NULL** **:** kolom tidak boleh bernilai *null*
2. **UNIQUE** : kolom hanya memiliki nilai tunggal
3. **PRIMARY KEY** **:** identifikasi unik untuk setiap baris pada tabel
4. **FOREIGN KEY** **:** hubungan kolom dengan kolom dari tabel referensi
5. **CHECK** : memberikan suatu kondisi yang bernilai benar

***Data Manipulation Language (*DML)**

Perintah yang termasuk dalah kategori DML adalah :

a. SELECT *nama_field* FROM *nama_tabel*
b. INSERT INTO *nama_tabel* (*field1,field2,…*) VALUES (*nilai1,nilai2,…*)
c. UPDATE *nama_tabel* SET *field1=nilai1,….*
d. DELETE FROM *nama_tabel* WHERE *field1=nilai1*

Command SQL :

Perintah	Keterangan
CREATE	Membuat tabel atau field
ALTER	Mengubah tabel dengan menambah field atau mengubah definisi field
DROP	Men-DROP tabel
SELECT	Mendefinisikan recordset, data apa yang akan ditampilkan dari database
INSERT	Menyisipkan recordset
UPDATE	Mengubah recordset
DELETE	Menghapus recordset

Ketika menggunakan ***query***, pengguna dapat menggunakan klausa berikut untuk diimplementasikan dalam ***statement SQL***. Klausa SQL:

Klausa	Keterangan
FROM	Menentukan tabel mana yang datanya akan ditampilkan
WHERE	Menentukan kondisi query
GROUP BY	Menentukan grup / kelompok dari informasi yang dipilih
HAVING	Digunakan bersama GROUP BY untuk menentukan kondisi untuk tiap grup dalam query
ORDER BY	Menentukan urutan *(sort)* data dari query

Ekspresi Aritmatika

Operator	Deskripsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian

Urutan Pengerjaan: *, /, +, -

Operator Perbandingan

Operator	Arti
=	Sama dengan
>	Lebih besar dari
>=	Lebih besar atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
<>	Tidak sama dengan
!=	Tidak sama dengan
^=	Tidak sama dengan
BETWEEN ... AND ...	Berada di antara 2 value
IN(*himpunan*)	Yang cocok dengan salah satu yang terdapat dalam set
LIKE	Yang cocok dengan pola karakter tertentu
IS NULL	Jika value-nya merupakan nilai null

Logika Kondisi

Operator	Arti
AND	Menghasilkan TRUE apabila kedua komponen benar
OR	Menghasilkan TRUE apabila salah satu komponen benar
NOT	Menghasilkan TRUE apabila kondisinya false

Tata Urutan Operator

- Dari sekian banyak operator yang telah disebutkan sebelumnya, tata urutan pengerjaannya dapat dilihat pada tabel berikut.

Urutan Pengerjaan	Operator
1	Operator aritmetika
2	Operator penggabungan
3	Operator perbandingan
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Kondisi logika NOT
7	Kondisi logika AND
8	Kondisi logika OR

Operator AND

a. AND digunakan sebagai kriteria “DAN”

b. Penggunaan : operand AND operand

Operator OR

a. OR digunakan sebagai kriteria “ATAU”

b. Penggunaan : operand OR operand

Operator NOT

a. Untuk menyatakan “TIDAK” atau “BUKAN”

b. Penggunaan : NOT kondisi

Operator BETWEEN

Operator BETWEEN untuk menyatakan suatu jangkauan.

Operator IN dan NOT IN

a. **IN** digunakan untuk melakukan pencocokan terhadap suatu daftar nilai.

b. **NOT IN** berarti “tidak cocok dengan”.

Operator LIKE dan NOT LIKE

a. **LIKE** digunakan untuk mencari data menurut awalan, akhiran atau penggalan suatu kata atau suku-kata.

b. Menambahkan tanda *wildcard* (_) atau tanda persen (%).

c. **NOT LIKE** digunakan untuk kebalikan dengan LIKE.

Wildcard untuk LIKE

- Tanda seperti % dinamakan ***wildcard***.
- *Wildcard* % Cocok dengan nol, satu atau sejumlah karakter apa saja. Huruf besar atau kecil dianggap sama.

Bentuk	Keterangan
%h%	Cocok dengan karakter apa saja yang mengandung karakter h atau H.
%g	Cocok dengan karakter yang berakhiran g atau G.
S%	Cocok dengan karakter yang berawalan s atau S.

Penggunaan ***wildcard*** :

- *Wildcard _ (underscore)* digunakan untuk mencocokkan dengan sebuah karakter apa saja.
- Misalnya, **a_u** cocok dengan **abu, alu, anu** atau **awu**. Tetapi tidak sesuai dengan **abru, altru** atau **accu**.

Contoh:

SELECT nama FROM mahasiswa WHERE nama LIKE “%i_a%";

Menampilkan nama yang mengandung i diikuti dengan sebuah huruf apa saja dan kemudian diikuti dengan a.

Klausa **DISTINCT** berguna untuk menampilkan data dengan menghilangkan yang kembar

LANGKAH PRAKTIKUM

1. Buat Tabel Mahasiswa dan isi data seperti berikut.

```
mysql> select * from mahasiswa;
+------+------------+--------------+--------------+------------+------+
| NIM  | Nama       | JenisKelamin | TanggalLahir | Alamat     | IPK  |
+------+------------+--------------+--------------+------------+------+
| H101 | Sita Putri | P            | 2000-04-18   | Pontianak  | 3.41 |
| H102 | Fitri Ade  | P            | 2000-06-10   | Pontianak  | 2.75 |
| H103 | Andrian    | L            | 1999-03-02   | Singkawang | 3.22 |
| H104 | Irawan     | L            | 1999-06-20   | Sanggau    | 3.58 |
| H105 | Rino Putra | L            | 1999-01-30   | Sintang    | 2.88 |
| H106 | Tyo Kurnia | L            | 2000-08-16   | Singkawang | 3.08 |
| H107 | Riani      | P            | 2000-09-12   | Pontianak  | 3.85 |
| H108 | Budianto   | L            | 2000-01-03   | Ketapang   |  3.5 |
| H109 | Rendra     | L            | 1999-04-06   | Sanggau    | 2.92 |
| H110 | Endah      | P            | 1999-07-12   | Pontianak  | 3.11 |
+------+------------+--------------+--------------+------------+------+
```

2. Gunakan operator Between untuk melihat nama dan tanggal lahir pada tabel mahasiswa yang memiliki tanggal lahir mulai dari tahun 1999 – 01 - 01 sampai 1999 – 12- 31.

```
mysql> select Nama, TanggalLahir FROM mahasiswa WHERE
    -> TanggalLahir BETWEEN '1999-01-01'AND
    -> '1999-12-31';
+------------+--------------+
| Nama       | TanggalLahir |
+------------+--------------+
| Andrian    | 1999-03-02   |
| Irawan     | 1999-06-20   |
| Rino Putra | 1999-01-30   |
| Rendra     | 1999-04-06   |
| Endah      | 1999-07-12   |
+------------+--------------+
```

3. Gunakan operator NOT Between untuk melihat nama dan tanggal lahir pada tabel mahasiswa yang tidak memiliki tanggal lahir mulai dari tahun 1999 – 01 - 01 sampai 1999 – 12- 31.

```
mysql> select Nama, TanggalLahir FROM mahasiswa WHERE
    -> TanggalLahir NOT BETWEEN '1999-01-01'AND
    -> '1999-12-31';
+------------+--------------+
| Nama       | TanggalLahir |
+------------+--------------+
| Sita Putri | 2000-04-18   |
| Fitri Ade  | 2000-06-10   |
| Tyo Kurnia | 2000-08-16   |
| Riani      | 2000-09-12   |
| Budianto   | 2000-01-03   |
+------------+--------------+
```

4. Gunakan operator IN untuk melihat nama dan alamat mahasiswa yang beralamat di Pontianak, Singkawang dan Sintang.

```
mysql> select Nama, Alamat FROM mahasiswa WHERE
    -> Alamat IN ('Pontianak','Singkawang','Sanggau');
+------------+------------+
| Nama       | Alamat     |
+------------+------------+
| Sita Putri | Pontianak  |
| Fitri Ade  | Pontianak  |
| Andrian    | Singkawang |
| Irawan     | Sanggau    |
| Tyo Kurnia | Singkawang |
| Riani      | Pontianak  |
| Rendra     | Sanggau    |
| Endah      | Pontianak  |
+------------+------------+
```

5. Gunakan operator NOT IN untuk melihat nama dan alamat mahasiswa yang tidak beralamat di Pontianak, Singkawang dan Sintang.

```
mysql> select Nama, Alamat FROM mahasiswa WHERE
    -> Alamat NOT IN ('Pontianak','Singkawang','Sanggau');
+------------+----------+
| Nama       | Alamat   |
+------------+----------+
| Rino Putra | Sintang  |
| Budianto   | Ketapang |
+------------+----------+
```

6. Gunakan Wildcard LIKE untuk melihat nama mahasiswa yang memiliki karakter yaitu karakter pertama R, karakter kedua bebas dan karakter ketiga A.

```
mysql> Select Nama from mahasiswa WHERE Nama LIKE '%R_A%';
+---------+
| Nama    |
+---------+
| Andrian |
| Riani   |
+---------+
```

7. Gunakan Wildcard LIKE untuk melihat nama mahasiswa yang memiliki karakter berakhiran E.

```
mysql> Select Nama from mahasiswa WHERE Nama LIKE '%E';
+-----------+
| Nama      |
+-----------+
| Fitri Ade |
+-----------+
```

8. Gunakan Wildcard LIKE untuk melihat nama mahasiswa yang memiliki karakter berawalan S.

```
mysql> Select Nama from mahasiswa WHERE Nama LIKE 'S%';
+------------+
| Nama       |
+------------+
| Sita Putri |
+------------+
```

9. Gunakan Distinct untuk melihat alamat mahasiswa yang ada pada tabel mahasiswa.

```
mysql> select DISTINCT Alamat From mahasiswa;
+------------+
| Alamat     |
+------------+
| Pontianak  |
| Singkawang |
| Sanggau    |
| Sintang    |
| Ketapang   |
+------------+
```

10. Gunakan Group By dan Having untuk melihat nama dan IPK mahasiswa yang memiliki IPK lebih dari 3.20.

```
mysql> Select Nama, IPK From mahasiswa GROUP BY Nama
    -> HAVING IPK>3.20;
+------------+------+
| Nama       | IPK  |
+------------+------+
| Andrian    | 3.22 |
| Budianto   |  3.5 |
| Irawan     | 3.58 |
| Riani      | 3.85 |
| Sita Putri | 3.41 |
+------------+------+
```