MODUL VI

Q u e r y

Tujuan :

Setelah menyelesaikan modul ini, anda diharapkan dapat :

1. Memilih data berdasarkan kriteria yang diinginkan.
2. Menampilkan data dengan kriteria tertentu.

Dasar Teori

Menampilkan data adalah hal yang sangat penting karena kita harus melihat dan menyeleksi suatu data dalam table maupun antar table. Untuk Melihat data atau Selection, Query yang digunakan adalah SELECT yang diikuti beberapa pernyataan khusus berkenaan dengan tabel yang diseleksi.

a. Mengambil data dengan SELECT

Data yang kita simpan dalam tabel dapat kita ambil menggunakan statement SELECT. Bentuk dasar statemen SELECT adalah sebagai berikut :

```
SELECT what_to_select
FROM wich_table
WHERE conditions_to_satisfy
```

what_to_select adalah informasi apa yang ingin kita lihat. Karakter (*) /bintang dapat menampilkan semua kolom.

wich_table menunjukkan dari tabel mana informasi tersebut akan diambil.

WHERE bersifat optional, diikuti dengan conditions_to_satisfy yang menunjukkan kondisi yang harus dipenuhi oleh sebuah baris informasi agar dapat dipilih. Contoh sebagai berikut :

```
mysql> select * from employee;
+----+------------+-----------+------------+------------+--------+-----------+-------------+
| id | first_name | last_name | start_date | end_date   | salary | city      | description |
+----+------------+-----------+------------+------------+--------+-----------+-------------+
|  1 | alison     | martin    | 1996-07-25 | 2006-07-25 | 1235.56 | toronto   | programmer  |
|    | alison     | mathews   | 1976-03-21 | 1986-02-21 | 6662.78 | vancouver | tester      |
|    | james      | smith     | 1978-12-12 | 1990-03-15 | 6545.78 | vancouver | tester      |
|  4 | celia      | rice      | 1982-10-24 | 1999-04-21 | 2345.78 | vancouver | manager     |
|  9 | James      | Bond      | 1982-04-21 | 2002-09-23 | 1235.56 | London    | Spy         |
| 10 | Hercules   | Pairot    | 1973-05-23 | 2001-08-09 | 4322.98 | Brussels  | Detective   |
| 11 | Lincoln    | Rhyme     | 1999-05-25 | 2011-07-13 | 3213.98 | New York  | Forensics   |
| 12 | Sherlock   | Holmes    | 1923-08-12 | 1945-07-21 | 4124.21 | London    | Detective   |
+----+------------+-----------+------------+------------+--------+-----------+-------------+
8 rows in set (0.00 sec)
```

Kita dapat mengambil beberapa kolom saja dari tabel employee untuk ditampilkan, sebagai berikut :

```
mysql> select first_name, last_name, city from employee;
+------------+-----------+-----------+
| first_name | last_name | city      |
+------------+-----------+-----------+
| alison     | martin    | toronto   |
| alison     | mathews   | vancouver |
| james      | smith     | vancouver |
| celia      | rice      | vancouver |
| James      | Bond      | London    |
| Hercules   | Pairot    | Brussels  |
| Lincoln    | Rhyme     | New York  |
| Sherlock   | Holmes    | London    |
+------------+-----------+-----------+
8 rows in set (0.00 sec)
```

b. Query menggunakan parameter kondisi WHERE

Statement WHERE dapat digunakan untuk memfilter data yang ingin kita ambil. Berikut adalah beberapa contoh penggunaan parameter kondisi WHERE.

```
mysql> select first_name, last_name, city from employee WHERE city = 'New York';
+------------+-----------+----------+
| first_name | last_name | city     |
+------------+-----------+----------+
| Lincoln    | Rhyme     | New York |
+------------+-----------+----------+
1 row in set (0.02 sec)
```

c. Query menggunakan beberapa parameter kondisional

Kita dapat memilih data menggunakan beberapa kombinasi parameter kondisional dihubungkan dengan ststement AND atau OR. Statement AND dapat juga ditulis sebagai '&&', sedangnkan statement OR juga dapat ditulis sebagai '||'. Statement AND memiliki **precedence yang lebih tinggi** dibandingkan dengan statement OR. Contoh menggunakan operator AND sebagai berikut :

```
mysql> select first_name, last_name, salary, city from employee WHERE city = 'Vancouver' AND salary > 4000;
+------------+-----------+---------+-----------+
| first_name | last_name | salary  | city      |
+------------+-----------+---------+-----------+
| alison     | mathews   | 6662.78 | vancouver |
| james      | smith     | 6545.78 | vancouver |
+------------+-----------+---------+-----------+
2 rows in set (0.00 sec)
```

Contoh penggunaan operator OR adalah sebagai berikut :

```
mysql> select first_name, last_name, city, description
    -> FROM employee
    -> WHERE city = 'vancouver' OR description = 'Spy';
+------------+-----------+-----------+-------------+
| first_name | last_name | city      | description |
+------------+-----------+-----------+-------------+
|     |on    | mathews   | vancouver | tester      |
|     |s     | smith     | vancouver | tester      |
| celia      | rice      | vancouver | manager     |
| James      | Bond      | London    | Spy         |
+------------+-----------+-----------+-------------+
4 rows in set (0.00 sec)
```

d. Memberikan alias hasil query pada SELECT

Kita dapat memberikan alias pada kolom hasil pencarian dengan menggunakan keyword AS. Berikut ini adalah contohnya :

```
mysql> SELECT CONCAT (first_name," ",last_name) AS name, description
    -> FROM employee
    -> WHERE description = "Detective";
+-----------------+-------------+
| name            | description |
+-----------------+-------------+
| Hercules Pairot | Detective   |
| Sherlock Holmes | Detective   |
+-----------------+-------------+
2 rows in set (0.00 sec)
```

Dari hasil di atas, dapat dilihat bahwa gabungan antara first_name dan last_name diberi nama alias ‘name’.

e. Query data dengan text dengan PATTERN MACHING

Pattern maching dapat kita gunakan untuk memilih data bertipe teks dengan karakteristik tertentu. Perintah yang digunakan untuk melakukan perncocokan adalah LIKE dan NOT LIKE. Berikut adalah beberapa fasilitas pattern maching yang disediakan oleh MySQL.

simbol	fungsi
-	Match any single character
%	Match an arbitrary number of character (including no character)

Berikut adalah contohnya :

```
mysql> select concat (first_name," ",last_name)
    -> from employee
    -> WHERE first_name LIKE 'J____';
+-------------------------------------+
| concat (first_name," ",last_name) |
+-------------------------------------+
| james smith                         |
| James Bond                          |
+-------------------------------------+
2 rows in set (0.00 sec)
```

Contoh di atas menunjukkan bagaimana memilih employee yang bernama depan diawali dengan huruf “J” dan diikuti oleh tepat 4 buah karakter apapun.

```
mysql> select concat (first_name," ",last_name)
    -> from employee
    -> WHERE first_name NOT LIKE '%n';
+-------------------------------------+
| concat (first_name," ",last_name) |
+-------------------------------------+
| james smith                         |
| celia rice                          |
| James Bond                          |
| Hercules Pairot                     |
| Sherlock Holmes                     |
+-------------------------------------+
5 rows in set (0.00 sec)
```

Contoh di atas menunjukkan bagaimana memilih employee yang bernama depan **tidak diakhiri** dengan karakter “n”.

f. Query data unik menggunakan DISTINC

DSITINC digunakan untuk menghilangkan duplikasi dari data yang dicari sehingga didapatkan data yang unik (hanya muncul satu kali). Berikut adalah contohnya :

```
mysql> select description from employee;
+-------------+
| description |
+-------------+
 |programmer
     |er
     |er
| manager     |
| Spy         |
| Detective   |
| Forensics   |
| Detective   |
+-------------+
8 rows in set (0.00 sec)
```

Tampilan di atas adalah data pada kolom ‘description’, terdapat 8 baris. Kemudian dilakukan SELECT DISTINCT, makahasilnya seperti di bawah ini, tersisa 6 baris, karena terdapat 2 description yang sama.

```
mysql> SELECT DISTINCT description FROM employee;
+-------------+
| description |
+-------------+
 |programmer
     |er
| manager     |
| Spy         |
| Detective   |
| Forensics   |
+-------------+
6 rows in set (0.00 sec)
```

g. Membatasi hasil query dengan LIMIT

Data yang dihasilkan dari query yang kita masukkan dapat kita batasi menggunakan statement LIMIT. Berikut ini contohnya :

```
mysql> select * from employee
    -> LIMIT 5;
+----+------------+-----------+------------+------------+---------+-----------+-------------+
| id | first_name | last_name | start_date | end_date   | salary  | city      | description |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
|  1 | alison     | martin    | 1996-07-25 | 2006-07-25 | 1235.56 | toronto   | programmer  |
|    | alison     | mathews   | 1976-03-21 | 1986-02-21 | 6662.78 | vancouver | tester      |
|    | james      | smith     | 1978-12-12 | 1990-03-15 | 6545.78 | vancouver | tester      |
|  4 | celia      | rice      | 1982-10-24 | 1999-04-21 | 2345.78 | vancouver | manager     |
|  9 | James      | Bond      | 1982-04-21 | 2002-09-23 | 1235.56 | London    | Spy         |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
5 rows in set (0.00 sec)
```

Pada contoh query di atas, kita membatasi jumlah data yang ditampilkan sebanyak lima data saja menggunakan statement LIMIT 5. Kita juga membatasi disertai pemilihan batasan tersebut ditampilkan mulai data keberapa, masih menggunakan statement LIMIT. Berikut ini adalah contohnya :

```
mysql> select * from employee
    -> LIMIT 2,3;
+----+------------+-----------+------------+------------+---------+-----------+-------------+
| id | first_name | last_name | start_date | end_date   | salary  | city      | description |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
|    | james      | smith     | 1978-12-12 | 1990-03-15 | 6545.78 | vancouver | tester      |
|  4 | celia      | rice      | 1982-10-24 | 1999-04-21 | 2345.78 | vancouver | manager     |
|  9 | James      | Bond      | 1982-04-21 | 2002-09-23 | 1235.56 | London    | Spy         |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
3 rows in set (0.00 sec)
```

Pada contoh query di atas, kita membatasi data yang ditampilkan dimulai dari data ke-2 sebanyak 3 data. Hal yang perlu diperhatikan, urutan data dimulai dari urutan ke-0. Sehingga jika kita menampilkan data menggunakan LIMIT 0,3 akan terlihat sebagai berikut :

```
mysql> select * from employee
    -> LIMIT 0,3;
+----+------------+-----------+------------+------------+---------+-----------+-------------+
| id | first_name | last_name | start_date | end_date   | salary  | city      | description |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
|  1 | alison     | martin    | 1996-07-25 | 2006-07-25 | 1235.56 | toronto   | programmer  |
|    | alison     | mathews   | 1976-03-21 | 1986-02-21 | 6662.78 | vancouver | tester      |
|    | james      | smith     | 1978-12-12 | 1990-03-15 | 6545.78 | vancouver | tester      |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
3 rows in set (0.00 sec)
```

h. Mengelompokkan hasil query dengan GROUP BY

Hasil query dapat kita kelompokkan berdasarkan *field/*kolom menggunakan statement GROUP BY. Berikut ini adalah contohnya :

```
mysql> select * from employee GROUP BY city;
+----+------------+-----------+------------+------------+---------+-----------+-------------+
| id | first_name | last_name | start_date | end_date   | salary  | city      | description |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
| 10 | Hercules   | Pairot    | 1973-05-23 | 2001-08-09 | 4322.98 | Brussels  | Detective   |
|  9 | James      | Bond      | 1982-04-21 | 2002-09-23 | 1235.56 | London    | Spy         |
| 11 | Lincoln    | Rhyme     | 1999-05-25 | 2011-07-13 | 3213.98 | New York  | Forensics   |
|  1 | alison     | martin    | 1996-07-25 | 2006-07-25 | 1235.56 | toronto   | programmer  |
|    | alison     | mathews   | 1976-03-21 | 1986-02-21 | 6662.78 | vancouver | tester      |
+----+------------+-----------+------------+------------+---------+-----------+-------------+
5 rows in set (0.00 sec)
```

Hasil di atas menunjukkan pengelompokan berdasarkan “city”. Hasil query menunjukkan data yang ditampilkan adalah data yang pertama untuk setiap “city”. Hal yang perlu diperhatikan adalah data yang ditampilkan terurut berdasarkan data pada kolom “city”.

i. Menggunakan COUNT

Query dengan menggunakan GRUOP BY hanya menunjukkan data pertama yang muncul. Jika kita ingin mendapatkan jumlah anggota setiap kelompok, maka kita dapat menggunakan fungsi COUNT(). Berikut ini contohnya :

```
mysql> select city, COUNT(*) FROM employee GROUP BY city;
+-----------+----------+
| city      | COUNT(*) |
+-----------+----------+
| Brussels  |        1 |
| London    |        2 |
| New York  |        1 |
| toronto   |        1 |
| vancouver |        3 |
+-----------+----------+
5 rows in set (0.00 sec)
```

Hasil query di atas menunjukkan jumlah employee di setiap kotanya. Kita juga dapat melakukan kombinasi GROUP BY dengan parameter kondisi sebagai berikut :

```
mysql> select city, COUNT(*)
    -> FROM employee
    -> WHERE description = 'Detective'
    -> GROUP BY city;
+----------+----------+
| city     | COUNT(*) |
+----------+----------+
| Brussels |        1 |
| London   |        1 |
+----------+----------+
2 rows in set (0.00 sec)
```

Menujukkan bahwa detective di kota Brussels terdapat 1 orang dan di kota London terdapat 1 orang.

j. Parameter kondisional dengan HAVING

Statemen having merupakan parameter koondisional seperti WHERE, yang bertindak sebagai pembatas sekunder dari hasil query. Statament HAVING biasanya digunakan untuk pembatas sekunder setelah statement GROUP BY, walaupun bisa saja digunakan tanpa menggunakan perintah GROUP BY. Berikut ini contohnya :

```
mysql> select first_name, last_name, salary
    -> from employee
    -> HAVING salary > 3000;
+------------+-----------+---------+
| first_name | last_name | salary  |
+------------+-----------+---------+
| alison     | mathews   | 6662.78 |
| james      | smith     | 6545.78 |
| Hercules   | Pairot    | 4322.98 |
| Lincoln    | Rhyme     | 3213.98 |
| Sherlock   | Holmes    | 4124.21 |
+------------+-----------+---------+
5 rows in set (0.00 sec)
```

Query di atas menunjukkan parameter HAVING dapat digunakan seperti parameter WHERE.

```
mysql> select city, COUNT(*), salary FROM employee WHERE salary > 3000 GROUP BY city;
+-----------+----------+---------+
| city      | COUNT(*) | salary  |
+-----------+----------+---------+
| Brussels  |        1 | 4322.98 |
| London    |        1 | 4124.21 |
| New York  |        1 | 3213.98 |
| vancouver |        2 | 6662.78 |
+-----------+----------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select city, COUNT(*), salary FROM employee GROUP BY city HAVING salary > 3000;
+-----------+----------+---------+
| city      | COUNT(*) | salary  |
+-----------+----------+---------+
| Brussels  |        1 | 4322.98 |
| New York  |        1 | 3213.98 |
| vancouver |        3 | 6662.78 |
+-----------+----------+---------+
3 rows in set (0.00 sec)
```

Contoh query di atas menunjukkan perbedaan urutan dijalankannya filtering, sehingga didapatkan data yang berbeda. Query yang pertama melakukan pemilihan salary > 3000 terlebih dahulu sebelum kemudian dikelompokkan berdasarkan salary. Sedangkan query kedua melakukan pengelompokkan terlebih dahulu terhadap salary. Pengelompokan tersebut menyebabkan data pertama untuk setiap kelompok yang terpilih. Ketika parameter HAVING dijalankan, query hanya akan menampilkan data pertama untuk setiap kelompok yang memiliki salary > 3000. Penggunaan HAVING setelah ORDER BY memerlukan nama kolom yang akan difilter menggunakan HAVING ikut dipilih. Jika tidak akan muncul pesan error seperti di bawah ini :

```
mysql> select city, COUNT(*) FROM employee GROUP BY city HAVING salary > 3000;
ERROR 1054 (42S22): Unknown column 'salary' in 'having clause'
```

k. Mengurutkan dengan ORDER BY

Hasil query dapa\t kita urutkan berdasarkan field/kolom tertentu menggunakan ORDER BY. Statement ASC dan DESC dapat kita gunakan untuk mendapatkan pengurutan naik atau turun. Berikut adalah contoh penggunaannya :

```
mysql> select concat (first_name," ",last_name) AS name
    -> from employee
    -> ORDER BY name;
+-----------------+
| name            |
+-----------------+
| alison martin   |
| alison mathews  |
| celia rice      |
| Hercules Pairot |
| James Bond      |
| james smith     |
| Lincoln Rhyme   |
| Sherlock Holmes |
+-----------------+
8 rows in set (0.01 sec)
```

l. Mengurutkan hasil query berdasarkan kolom

Mengurutkan hasil query dapat diurutkan berdasarkan lebih dari satu kolom. Statement urutan (ASC dan DESC) melekat pada kolom yang mendahuluinya. Berikut contohnya :

```
mysql> select  first_name, last_name, city from employee ORDER BY first_name, city;
+------------+-----------+-----------+
| first_name | last_name | city      |
+------------+-----------+-----------+
| alison     | martin    | toronto   |
| alison     | mathews   | vancouver |
| celia      | rice      | vancouver |
| Hercules   | Pairot    | Brussels  |
| James      | Bond      | London    |
| james      | smith     | vancouver |
| Lincoln    | Rhyme     | New York  |
| Sherlock   | Holmes    | London    |
+------------+-----------+-----------+
8 rows in set (0.02 sec)
```

Contoh query tersebut menunjukkan pengurutan berdasarkan first_name terlebih dahulu sebelum mengurutkan berdasarkan city. Jika terdapat data dengan first_name yang sama, maka data tersebut akandiurutkan berdasarkan city.

m. Kombinasi ORDER BY dengan LIMIT

Ketika statement ORDER BY dikombinasikan dengan LIMIT, maka statement ORDER BY yang akan dijalankan terlebih dahulu, baru kemudian LIMIT dilakukan untuk membatasi jumlah hasil query yang ditampilkan. Contoh sebagai berikut :

```
mysql> select  first_name, last_name, city from employee ORDER BY city limit 4;
+------------+-----------+----------+
| first_name | last_name | city     |
+------------+-----------+----------+
| Hercules   | Pairot    | Brussels |
| James      | Bond      | London   |
| Sherlock   | Holmes    | London   |
| Lincoln    | Rhyme     | New York |
+------------+-----------+----------+
4 rows in set (0.00 sec)
```

n. Operator BETWEEN

Operator BETWEEN dibunakan untuk memfilter data yang bernilai di antara dua buah nilai yang dispesifikasikan. Berikut contohnya :

```
mysql> select  first_name, last_name, salary from employee
    -> WHERE salary BETWEEN 1000 and 3000;
+------------+-----------+---------+
| first_name | last_name | salary  |
+------------+-----------+---------+
| alison     | martin    | 1235.56 |
| celia      | rice      | 2345.78 |
| James      | Bond      | 1235.56 |
+------------+-----------+---------+
3 rows in set (0.00 sec)
```

Praktikum

Buatlah tutorial untuk tema di atas.

Tugas

Buatlah dokumentasi hasil praktikum yang anda lakukan dengan menggunakan note atau catatan atau dengan perintah \T pada *command line* anda, letakkan pada drive 'd:\namaanda\note\tgsmodul6.doc'. gunakan komentar untuk mempermudah memberikan catatan pada setiap kelompok perintah dengan menggunakan perintah # atau --.