BAB VI
SELEKSI TABEL

Satu hal fungsi terpenting yang harus ada dalam RDBMS yaitu dalam mengakses data yang ada dalam tabel di sistem. Fungsi yang disediakan dalam MYSQL yaitu fungsi SELECT yang berguna dalam mengakses data baik secara kolom maupun baris, dari satu tabel maupun lebih. Ketika anda megeksekusi fungsi tersebut, maka fungsi tersebut akan mengeluarkan berupahasil yang kita butuhkan. Dalam hal ini kita akan belajar tentang :

1. Mengetahui cara membuat statement yang mengandung informasi seluruh isi tabel maupun spesifik tabel.
2. Menambahkan fungsi optional pada statement SELECT.
3. Menambahkan clausa pilihan pada statement SELECT yang membatasi baik dari baris,group, dan lainnya.

VI.1 SELECT statement

Fungsi SELECT merupakan fungsi yang sering digunakan. Fungsi ini dapat memberikan hasil berupa data yang ada pada tabel baik berupa baris maupun kolom. Dibawah ini merupakan syntax SELECT pada MYSQL :

```
<select statement>::=
SELECT
[<select option> [<select option>...]]
{* | <select list>}
[<export definition>]
[
FROM <table reference> [{, <table reference>}...]
[WHERE <expression> [{<operator> <expression>}...]]
[GROUP BY <group by definition>]
[HAVING <expression> [{<operator> <expression>}...]]
[ORDER BY <order by definition>]
[LIMIT [<offset>,] <row count>]
[PROCEDURE <procedure name> [(<argument> [{, <argument>}...])]]
[{FOR UPDATE} | {LOCK IN SHARE MODE}]
]
<select option>::=
```

{ALL | DISTINCT | DISTINCTROW}

| HIGH_PRIORITY

| {SQL_BIG_RESULT | SQL_SMALL_RESULT}

| SQL_BUFFER_RESULT

| {SQL_CACHE | SQL_NO_CACHE}

| SQL_CALC_FOUND_ROWS

| STRAIGHT_JOIN

<select list>::=

{<column name> | <expression>} [[AS] <alias>]

[{, {<column name> | <expression>} [[AS] <alias>]}...]

<export definition>::=

INTO OUTFILE '<filename>' [<export option> [<export option>]]

| INTO DUMPFILE '<filename>'

<export option>::=

{FIELDS

[TERMINATED BY '<value>']

[[OPTIONALLY] ENCLOSED BY '<value>']

[ESCAPED BY '<value>']}

| {LINES

[STARTING BY '<value>']

[TERMINATED BY '<value>']}

<table reference>::=

<table name> [[AS] <alias>]

[{USE | IGNORE | FORCE} INDEX <index name> [{, <index name>}...]]

<group by definition>::=

<column name> [ASC | DESC]

[{, <column name> [ASC | DESC]}...]

[WITH ROLLUP]

<order by definition>::=

<column name> [ASC | DESC]

[{, <column name> [ASC | DESC]}...]

Dilihat dari syntax yang diatas, beberapa element terdapat dalam fungsi SELECT. Beberapa fungsi tambahan dalam SELECT memiliki kegunaan masing-masing.

Jika dilihat dari syntax :

```
SELECT
{* | <select list>}
```

<select list> mengandung clausa-clausa/ ekspresi untuk menampilkan kolom yang diinginkan. Secara syntax isi <select list>seperti ini :

```
<select list>::=
{<column name> | <expression>} [[AS] <alias>]
[{, {<column name> | <expression>} [[AS] <alias>]}...]
```

Seperti yang dijelaskan diatas bahwa minimal terdapat satu clausa pada fungsi SELECT. Jika terdapat lebih dari satu clausa,pembatasnya diberi tanda koma(,). Pada ekspresi itu juga dapat diberi alias yang dapat digunakan pada fungsi SELECT lainnya. Clausa SELECT hanya bisa digunakan pada fungsi SELECT dan data belum bisa dieksekusi sebelum mengisi nama tabel setelah clausa FROM. Clausa dapat diisi lebih dari satu tabel.

Inilah syntax pada FROM nya:

```
FROM <table reference> [{, <table reference>}...]
<table reference>::=
<table name> [[AS] <alias>]
[{USE | IGNORE | FORCE} INDEX <index name> [{, <index name>}...]]
```

Untuk lebih jelasnya kita akan masuk pada contoh berikut :

```
CREATE TABLE Angkot
(
 AngkotID  SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 NamaSupir VARCHAR(50) NOT NULL,
 Jurusan VARCHAR(50) NOT NULL,
 JenisMobil ENUM ('Toyota ','Daihatsu') NOT NULL,
 JumlahMobil INT(2) NOT NULL,
 MaxPenumpang  INT(2) NOT NULL,
 TahunAktif YEAR,
```

```
DataUpdate TIMESTAMP NOT NULL
);
```

Kemudian kita akan memasukkan datanya

```
INSERT INTO Angkot (NamaSupir, Jurusan, JenisMobil, JumlahMobil,
MaxPenumpang,TahunAktif )
VALUES ( "Adi","Cicaheum-Ciroyom",'Toyota',4,9,1990),
( "Ari","Dago-Kelapa",'Toyota',6,7,1995),
( "Andi","Cicaheum-Ciroyom",'Daihatsu',10,4,1996),
( "Budi","Dago-Kelapa",'Daihatsu',3,7,1989),
( "Cecep","Kelapa-Ledeng",'Toyota',9,8,1997),
( "Dika","Kalapa-Ledeng",'Daihatsu',5,6,1986),
( "Edi","Sadangserang-Caringin",'Toyota',4,8,1991),
( "Eka","Sadangserang-Caringin",'Daihatsu',5,9,1988);
```

Setelah data telah masuk dalam tabel,kita akan mencoba untuk memeriksa pakah data yang telah kita masukkan tadi sudah ada atau tidak,yaitu dengan cara menjalankan statement berikut :

```
SELECT * FROM Angkot ;
```

Penjelasan :

SELECT <' * ' menandakan semua data pada tabel akan ditampilkan > FROM <nama tabel> ;

Dan data yang akan ditampilkan setelah di eksekusi adalah :

```
+----------+-----------+----------------------+------------+-------------+--------------+------------+---------------------+
| AngkotID | NamaSupir | Jurusan              | JenisMobil | JumlahMobil | MaxPenumpang | TahunAktif |
DataUpdate        |
+----------+-----------+----------------------+------------+-------------+--------------+------------+---------------------+
|      1 | Adi      | Cicaheum-Ciroyom     | Toyota     |         4 |          9 |      1990 | 2009-12-30 21:55:28 |
|      2 | Ari      | Dago-Kelapa          | Toyota     |         6 |          7 |      1995 | 2009-12-30 21:55:28 |
|      3 | Andi     | Cicaheum-Ciroyom     | Daihatsu   |        10 |          4 |      1996 | 2009-12-30 21:55:28 |
|      4 | Budi     | Dago-Kelapa          | Daihatsu   |         3 |          7 |      1989 | 2009-12-30 21:55:28 |
|      5 | Cecep    | Kelapa-Ledeng        | Toyota     |         9 |          8 |      1997 | 2009-12-30 21:55:28 |
|      6 | Dika     | Kalapa-Ledeng        | Daihatsu   |         5 |          6 |      1986 | 2009-12-30 21:55:28 |
|      7 | Edi      | Sadangserang-Caringin | Toyota     |         4 |          8 |      1991 | 2009-12-30 21:55:28 |
|      8 | Eka      | Sadangserang-Caringin | Daihatsu   |         5 |          9 |      1988 | 2009-12-30 21:55:28 |
+----------+-----------+----------------------+------------+-------------+--------------+------------+---------------------+
```

8 rows in set (0.00 sec)

Secara umum, data yang ada pada tabel akan selalu berubah dan memiliki data yang banyak. Jika kita hanya memrlukan beberapa berdasarkan kolomnya (AngkotID, NamaSupir,Jurusan,JenisMobil, JumlahMobil, MaxPenumpang, TahunAktif, DataUpdate), maka perintahnya sbb :

```
SELECT AngkotID,  Jurusan, JenisMobil FROM Angkot ;
```

Penjelasan :

SELECT <kolom pada tabel yang akan ditampilkan> FROM <nama tabel> ;

Dan jika di eksekusi maka tampilannya :

```
+----------+----------------------+------------+
| AngkotID | Jurusan              | JenisMobil |
+----------+----------------------+------------+
|        1 | Cicaheum-Ciroyom     | Toyota     |
|        2 | Dago-Kelapa          | Toyota     |
|        3 | Cicaheum-Ciroyom     | Daihatsu   |
|        4 | Dago-Kelapa          | Daihatsu   |
|        5 | Kelapa-Ledeng        | Toyota     |
|        6 | Kalapa-Ledeng        | Daihatsu   |
|        7 | Sadangserang-Caringin | Toyota    |
|        8 | Sadangserang-Caringin | Daihatsu  |
+----------+----------------------+------------+
8 rows in set (0.00 sec)
```

Adapun pengunaan alias pada fungsi SELECT yaitu sbb :

```
SELECT NamaSupir  AS Supir,  Jurusan AS Tujuan FROM Angkot ;
```

Penjelasan :

SELECT <nama kolom pada tabel> AS <nama kolom yg baru> FROM <nama tabel>

Dan jika di eksekusi maka tampilannya :

```
+-------+----------------------+
| Supir | Tujuan               |
+-------+----------------------+
| Adi   | Cicaheum-Ciroyom     |
```

Ari	Dago-Kelapa
Andi	Cicaheum-Ciroyom
Budi	Dago-Kelapa
Cecep	Kelapa-Ledeng
Dika	Kalapa-Ledeng
Edi	Sadangserang-Caringin
Eka	Sadangserang-Caringin

8 rows in set (0.05 sec)

Mengunakan Ekspresi dalam fungsi SELECT

Pada ekspresi ini, fungsi SELECT dapat dikombinasi dengan operator(aritmathic),nama kolom dan fungsi lainnya.

Untuk lebih lengkapnya seperti contoh berikut :

```
SELECT NamaSupir , JumlahMobil + MaxPenumpang  AS Total FROM Angkot ;
```

Penjelasan :

SELECT < jumlah antar 2 kolom yang memiliki jenis yang sama > AS <nama kolom baru> FROM <nama tabel> ;

Dan jika di eksekusi maka tampilannya :

NamaSupir	Total
Adi	13
Ari	13
Andi	14
Budi	10
Cecep	17
Dika	11
Edi	12
Eka	14

8 rows in set (0.00 sec) Kolom Total terbentuk dari 2 kolom JumlahMobil dan MaxPenumpang. Ekspresi diatas dapat dikombinasikan lebih dari dua variabel (kolom). Oleh karena itu kombinasi ini dapat digunakan sesuai kebutuhan yang ada.

Menggunakan variabel langsung pada SELECT statement

Fungsi ini merupakan cara lain dalam menggantikan variabel kolom. Secara syntax yaitu :

@<variable name>:={<column name> | <expression>} [[AS] <alias>]

Contohnya :

```
SELECT @Supir:=NamaSupir, @Tujuan:=Jurusan
FROM Angkot
WHERE TahunAktif= 1991;
```

Penjelasan :

SELECT @<nama variabel><" := " titik dua(:) dan sama dengan (=) ><nama variabel baru>

FROM <nama tabel>

WHERE <kolom tabel kondisi>;

Fungsi ini sama dengan fungsi AS yang menggantikan nama kolom. Jadi penggunaannya disesuaikan menurut keperluan.

Menggunakan statement SELECT untuk menampilkan nilai

Contoh ini menampilkan fungsi SELECT yang lain, sehingga tidak hanya berpatokan pada contoh yang ada sebelumnya.

```
SELECT 1+3, 'CD Inventory', NOW() AS 'Date/Time';
```

Penjelasan :

SELECT <merupakan perhitungan sesuai elementnya>, <list dari elemennya>, <fungsi waktu>;

Pilihan / optional statement SELECT

Pada saat membuat fungsi SELECT, clausa SELECT dapat digunakan lebih dari satu. Adapun syntaxnya sebagai berikut:

<select option>::=
{ALL | DISTINCT | DISTINCTROW}
| HIGH_PRIORITY
| {SQL_BIG_RESULT | SQL_SMALL_RESULT}
| SQL_BUFFER_RESULT
| {SQL_CACHE | SQL_NO_CACHE}
| SQL_CALC_FOUND_ROWS
| STRAIGHT_JOIN

Untuk lebih jelasnya,kita akan implementasikan :

```
SELECT ALL Jurusan , JenisMobil
FROM Angkot;
```

Penjelasan :

SELECT ALL <nama - nama kolom>

FROM <nama tabel>;

Kegunaannya yaitu menampilkan semua data yang ada pada jurusan dan jenis mobil,walaupun datanya banyak yang sama dan yang ditampilkan semua data yang ada pada kolom tersebut. Hasilnya sebagai berikut

```
+----------------------+------------+
| Jurusan              | JenisMobil |
+----------------------+------------+
| Cicaheum-Ciroyom     | Toyota     |
| Dago-Kelapa          | Toyota     |
| Cicaheum-Ciroyom     | Daihatsu   |
| Dago-Kelapa          | Daihatsu   |
| Kelapa-Ledeng        | Toyota     |
| Kalapa-Ledeng        | Daihatsu   |
| Sadangserang-Caringin | Toyota    |
| Sadangserang-Caringin | Daihatsu  |
```

```
+----------------------+------------+
8 rows in set (0.01 sec)
```

```
SELECT DISTINCT Jurusan , JenisMobil
FROM Angkot;
```

Penjelasan :

SELECT ALL <nama - nama kolom>
FROM <nama tabel>;

Kegunaannya yaitu menampilkan semua data yang ada pada jurusan dan jenis mobil,serta data yang ditampilkan tidak ada yang sama untuk mengurangi duplikasi.. Hasilnya sebagai berikut :

```
+----------------------+------------+
| Jurusan              | JenisMobil |
+----------------------+------------+
| Cicaheum-Ciroyom     | Toyota     |
| Dago-Kelapa          | Toyota     |
| Cicaheum-Ciroyom     | Daihatsu   |
| Dago-Kelapa          | Daihatsu   |
| Kelapa-Ledeng        | Toyota     |
| Kalapa-Ledeng        | Daihatsu   |
| Sadangserang-Caringin | Toyota    |
| Sadangserang-Caringin | Daihatsu  |
+----------------------+------------+
8 rows in set (0.01 sec)
```

VI.2 Clausa WHERE

Clausa WHERE digunakan untuk membuat satu atau lebih kondisi pada SELECT statement. Secara syntax adalah sebagai berikut :

```
WHERE <expression> [{<operator> <expression>}...]
```

Jika dilihat dari syntaxnya maka minimal kondisi yang ada yaitu satu kondisi. Dan jika terdapat lebih dari satu kondisi maka kondisi-kondisi tersebut harus saling terhubung dengan fungsi AND atau OR.

Sekarang kita lihat contohnya :

```
SELECT NamaSupir, Jurusan
FROM Angkot
WHERE TahunAktif= 1991;
```

Penjelasan:

SELECT <nama kolom> FROM <nama tabel>

WHERE <kondisi >;

Pada contoh ini data yang akan ditampilkan berdasarkan tahunaktif yang telah dispesifikkan yaitu tahun 1991. Kondisi ini bisa dirubah sesuai dengan penggunaaannya.

Inilah hasilnya :

```
+-----------+----------------------+
| NamaSupir | Jurusan              |
+-----------+----------------------+
| Edi       | Sadangserang-Caringin |
+-----------+----------------------+
1 row in set (0.00 sec)
```

Dan inilah contoh untul multiple condition

```
SELECT NamaSupir, Jurusan
FROM Angkot
WHERE TahunAktif= 1989 AND NamaSupir =”Budi”;
```

Inilah hasilnya :

```
+-----------+-------------+
| NamaSupir | Jurusan     |
+-----------+-------------+
```

```
| Budi     | Dago-Kelapa |
+----------+-------------+
1 row in set (0.00 sec)
```

Fungsi clausa GROUP BY

Fungsi group by ini minimal terdiri dari satu kolom yang terdefinisi. Fungsi GROUP ini digunakan untuk mengelompokan kolom-kolom berdsarkan urutan kolom yang telah terdefinisi pada tabel. Dalam fungsi ini jjg dapat diataur baik secra ascending maupun descending. Inilah syntax secar umumnya

```
GROUP BY <group by definition>
<group by definition>::=
<column name> [ASC | DESC]
[{, <column name> [ASC | DESC]}...]
[WITH ROLLUP]
```

Untuk lebih jelasnya kita lihat contoh berikut :

```
SELECT Jurusan, COUNT(*) AS Total
FROM Angkot
WHERE JenisMobil ='Toyota'
GROUP BY Jurusan;
```

Penjelasan :

SELECT <nama kolom pilihan>, < menghitung total baris pada baris kolom yang dipilih> AS Total

FROM <nama tabel>

WHERE <kondisi >

GROUP BY <kolom pilihan>;

Hasil yang akan diperoleh yaitu data yang akan ditampilkan berdasarkan gruop yang telah dipilih. Inilah hasilnya

```
+----------------------+-------+
| Jurusan              | Total |
+----------------------+-------+
| Cicaheum-Ciroyom     |     1 |
| Dago-Kelapa          |     1 |
| Kelapa-Ledeng        |     1 |
| Sadangserang-Caringin |     1 |
+----------------------+-------+
```

4 rows in set (0.05 sec)

Inilah contoh untuk multiple group

```
SELECT Jurusan, JenisMobil, COUNT(*) AS Total
FROM Angkot
GROUP BY Jurusan,JenisMobil;
```

Maka hasilnya :

```
+----------------------+------------+-------+
| Jurusan              | JenisMobil | Total |
+----------------------+------------+-------+
| Cicaheum-Ciroyom     | Toyota     |     1 |
| Cicaheum-Ciroyom     | Daihatsu   |     1 |
| Dago-Kelapa          | Toyota     |     1 |
| Dago-Kelapa          | Daihatsu   |     1 |
| Kalapa-Ledeng        | Daihatsu   |     1 |
| Kelapa-Ledeng        | Toyota     |     1 |
| Sadangserang-Caringin | Toyota     |     1 |
| Sadangserang-Caringin | Daihatsu   |     1 |
+----------------------+------------+-------+
```

8 rows in set (0.00 sec)

VI.3 Clausa HAVING

Fungsi clausa HAVING hampir sama dengan clausa WHERE. Pada fungsi ini dapet memasukkan aggregate. Fungsi agregate aalah fungsi untuk menghasilkan summarize seperti fungsi COUNT(). Syntax secara umum yaitu

HAVING <expression> [{<operator> <expression>}...]

Untuk memahaminya, kita langsung ke contohnya

```
SELECT Jurusan, COUNT(*) AS Total
FROM Angkot
WHERE JenisMobil='Toyota'
GROUP BY Jurusan
HAVING Total <2;
```

Data yang akan tampil yaitu data yang memiliki nilai Total yang kurag dari 10. Dan inilah hasilnya

```
+----------------------+-------+
| Jurusan              | Total |
+----------------------+-------+
| Cicaheum-Ciroyom     |     1 |
| Dago-Kelapa          |     1 |
| Kelapa-Ledeng        |     1 |
| Sadangserang-Caringin |     1 |
+----------------------+-------+
4 rows in set (0.00 sec)
```

VI.4 Clausa ORDER BY

Clausa ORDER BY pada statement SELECT

Secara umum syntaxnya

ORDER BY <order by definition>

<order by definition>::=

<column name> [ASC | DESC]

[{, <column name> [ASC | DESC]}...]

Paada syntax dapat dilihat bahwa minimal terdapat satu kolom. Jika terdapat lebih dari satu kolom maka sebagai pembatasnya maka harus diberi koma(,). Dan inilah contoh ORDER BY

```
SELECT NamaSupir , JenisMobil, MaxPenumpang
FROM Angkot
WHERE MaxPenumpang >6
ORDER BY NamaSupir DESC;
```

Hasilnya yaitu data yang akan ditampilkan akan diurutkan berdasarkan NamaSupir menurut ukuran alphabet paling akhir ke awal. Inilah tampilannya

```
+-----------+------------+--------------+
| NamaSupir | JenisMobil | MaxPenumpang |
+-----------+------------+--------------+
| Eka       | Daihatsu   |            9 |
| Edi       | Toyota     |            8 |
| Cecep     | Toyota     |            8 |
| Budi      | Daihatsu   |            7 |
| Ari       | Toyota     |            7 |
| Adi       | Toyota     |            9 |
+-----------+------------+--------------+
6 rows in set (0.00 sec)
```

Adapun fungsi ORDER BY dengan 2 kolom

```
SELECT NamaSupir , JenisMobil, MaxPenumpang
FROM Angkot
WHERE MaxPenumpang >6
ORDER BY NamaSupir DESC, JenisMobil ASC;
```

Dan hasilnya yatu

```
+-----------+------------+--------------+
| NamaSupir | JenisMobil | MaxPenumpang |
```

```
+-----------+------------+--------------+
| Eka       | Daihatsu   |            9 |
| Edi       | Toyota     |            8 |
| Cecep     | Toyota     |            8 |
| Budi      | Daihatsu   |            7 |
| Ari       | Toyota     |            7 |
| Adi       | Toyota     |            9 |
+6 rows in set (0.00 sec)
-----------+------------+--------------+
6 rows in set (0.00 sec)
```

VI.5 Clausa LIMIT

Claus a LIMIT sama dengan fungsi ORDER BY namun fungsi ini akan lebih efektif pada statement SELECT.Clausa LIMIT menggunakan 2 argumen dan secara umum syntaxnya

LIMIT [<offset>,] <row count>

Pilihan pertama<offset> merupakan pilihan optional. Nilai defaultnya 0. Dan pada argument kedua<row count> menunjukan jumlah aris yang akan ditampilkan.
Inilah contohnya

```
SELECT NamaSupir , JenisMobil, MaxPenumpang
FROM Angkot
WHERE MaxPenumpang=6
ORDER BY NamaSupir DESC
LIMIT 2;
```

Dan hasilnya

```
+-----------+------------+--------------+
| NamaSupir | JenisMobil | MaxPenumpang |
+-----------+------------+--------------+
| Dika      | Daihatsu   |            6 |
```

```
+-----------+------------+--------------+
```

1 row in set (0.00 sec)

Dan dibawah ini merupakan fungsi LIMIT dengan Offset

```
SELECT NamaSupir , JenisMobil, MaxPenumpang
FROM Angkot
WHERE MaxPenumpang >6
ORDER BY NamaSupir DESC, MaxPenumpang ASC
LIMIT 3,4;
```

Dan hasilnya

```
+-----------+------------+--------------+
| NamaSupir | JenisMobil | MaxPenumpang |
+-----------+------------+--------------+
| Budi      | Daihatsu   |            7 |
| Ari       | Toyota     |            7 |
| Adi       | Toyota     |            9 |
+-----------+------------+--------------+
```

3 rows in set (0.00 sec)