



Versi 1.0 | kenalicorp | 2016

Data Manipulation Language

Pendekatan Teori dan Praktik

Eko Win Kenali

POLITEKNIK NEGERI LAMPUNG

DML / Data Manipulation Language

DML merupakan perintah SQL yang berhubungan dengan manipulasi atau pengolahan data atau record dalam table. Perintah SQL yang termasuk dalam DML antara lain :

INSERT**MERGE - UPSERT****SELECT****CALL****UPDATE****EXPLAIN PLAN****DELETE****LOCK TABLE**

SELECT

Perintah SQL yang digunakan untuk menampilkan (SELECT/READ) data atau record dalam table. Baik keseluruhan record ataupun terbatas pada record yang memenuhi kriteria/kondisi tertentu

Sintak :

```
SELECT  
[ ALL | DISTINCT ]  
[ TOP ( expression ) [ PERCENT ] [ WITH TIES ] ]  
select_list [ INTO new_table ]  
FROM TableName  
[ WHERE [ search_condition ] ]  
[ GROUP BY group_by_expression ]  
[ HAVING search_condition ]  
[ ORDER BY order_expression [ ASC | DESC ] ]
```

SELECT

Contoh:

```
SELECT * FROM MAHASISWA;
```

```
SELECT * FROM MAHASISWA ORDER BY npm DESC;
```

```
SELECT DISTINCT tempat_lhr FROM MAHASISWA;
```

```
SELECT tempat_lhr FROM MAHASISWA GROUP BY tempat_lhr;
```

```
SELECT tempat_lhr FROM MAHASISWA GROUP BY tempat_lhr  
HAVING tempat_lhr = 'Bandar Lampung';
```

SQL

Mengenal Dasar Query Satu Tabel

Pendahuluan

- Bab ini akan membahas penggunaan pernyataan **select** untuk menampilkan isi sebuah tabel baik secara keseluruhan ataupun terbatas pada baris-baris yang memenuhi kriteria/kondisi tertentu.

Stemen Select

SELECT statement ::=

< query_expression >
[ORDER BY { *order_by_expression* | *column_position* [ASC | DESC] } [,...*n*]]
[COMPUTE { { AVG | COUNT | MAX | MIN | SUM } (*expression*) } [,...*n*] [BY *expression* [,...*n*]]]

< query expression > ::=
{ < query specification > | (< query expression >) }
[UNION [ALL] < query specification | (< query expression >) [...*n*]]

< query specification > ::=
SELECT [ALL | DISTINCT]
[**FROM** { < table_source > } [,...*n*]]
[WHERE < search_condition >]
[GROUP BY [ALL] *group_by_expression* [,...*n*] [WITH { CUBE | ROLLUP }]]
[HAVING < search_condition >]

Operator

Pada penggunaan pernyataan SELECT, operator-operator yang dapat terlibat :

- DISTINCT
- RELASIONAL
- LOGIKA
- BETWEEN dan NON BETWEEN
- IN dan NOT IN
- LIKE dan NOT LIKE
- IS NULL dan IS NOT NULL

Bahan

Untuk dapat menggunakan perintah-perintah SQL, di perlukan bahan berupa tabel-tabel. Berikut ini adalah struktur tabel yang di perlukan.

pegawai

	Column Name	Data Type	Length	Allow Nulls
▶	nip	nvarchar	10	
	nama	nvarchar	50	✓
	sex	char	1	✓
	tempat_lahir	nvarchar	30	✓
	tgl_lahir	datetime	8	✓
	tmt	datetime	8	✓
	kd_golongan	nvarchar	2	✓

sex

	Column Name	Data Type	Length	Allow Nulls
▶	sex	char	1	
	jenis_kelamin	nvarchar	15	✓
	sebutan	nvarchar	10	✓

golongan

	Column Name	Data Type	Length	Allow Nulls
▶	kd_golongan	nvarchar	2	
	nm_golongan	nvarchar	50	✓
	gapok	money	8	✓

pegawai

nip	nama	sex	tempat_lahir	tgl_lahir	tmt	Kd_golongan
132310741	Budiman	L	Lampung	02/01/1983	02/01/2004	01
132310742	Eko Win Kenali	L	Yogyakarta	02/09/1976	02/01/2004	03
132310743	Shinta Rahma	P	Lampung	12/01/1984	02/01/2004	02
132310744	Indah Muniarti	P	Jakarta	22/09/1990	02/01/2004	02
132310745	Heri Herlambang	L	Jakarta	15/05/1988	02/01/2004	01
132310739	Riko Hendrawan	L	Bandung	11/05/1976	02/01/2004	03
132310737	Kurniawan	L	Palembang	12/11/1972	02/01/2004	02
132310740	Sri Karmila	P	Jakarta	04/09/1991	02/01/2004	02
132310746	Dwi Arini	P	Lampung	11/11/1975	02/01/2004	01
132310747	Kamyono	L	Padang	04/09/1989	02/01/2004	01

golongan

Kd_golongan	Nm_golongan	gapok
01	II/A	600000
02	II/B	700000
03	III/A	850000
04	III/B	910000

sex

sex	Jenis_kelamin	sebutan
L	Laki-Laki	Mr.
P	Perempuan	Miss.

Percobaan-1

1. Menampilkan semua Kolom dan semua baris

```
SELECT * FROM nama_tabel;
```

- a. Percobaan :
Tampilkan semua data pada tabel pegawai
Tampilkan semua data pada tabel sex
Tampilkan semua data pada tabel golongan
2. Menampilkan Kolom tertentu pada sebuah tabel

```
SELECT nama_kolom_1,...nama_kolom_n FROM nama_tabel;
```

- a. Percobaan :
Tampilkan kolom nip, nama dan sex pada tabel pegawai
Tampilkan kolom jenis_kelamin dan sebutan pada tabel sex
Tampilkan kolom nm_golongan pada tabel golongan

Percobaan-1

3. Memperoleh data yang Unik

Data unik berarti tidak ada data yang kembar.

```
SELECT DISTINCT nama_kolom FROM nama_tabel;
```

- a. Percobaan :
- Sebelum menggunakan perintah tersebut, cobalah gunakan perintah berikut : **SELECT sex FROM pegawai;**
Amatilah dan catat hasilnya.
Gunakan operator DISTINCT untuk memperoleh nilai unik dari kolom sex pada tabel pegawai.
Amatilah dan Catat hasilnya

Percobaan-1

4. Memilih Baris/Record dengan kriteria Tertentu

Klausa yang digunakan adalah **WHERE**

```
SELECT nama_kolom FROM nama_tabel WHERE kondisi;
```

Kondisi : berupa suatu ekspresi logika dengan yang menyertakan lambang <,>,<=, >=, =, dan <>.

Contoh :

Menampilkan kolom **nip**, **nama** dan **sex** dari tabel **pegawai**, hanya jika pada kolom **sex** bernilai = 'L'.

```
SELECT nip,nama,sex FROM pegawai WHERE sex='L';
```

- a. Percobaan :
- Tampilkan semua kolom dan baris pada tabel pegawai jika kolom **kd_golongan** = '01'.
 - Tampilkan kolom **nama** dan **sex** dan baris pada tabel pegawai jika kolom **nama** = 'budiman'.
- Amati dan catat hasil kedua percobaan tersebut.

Percobaan-1

5. Memilih Baris/Record Tertentu menggunakan operator Relasional

```
SELECT nama_kolom FROM nama_tabel  
WHERE kondisi_1 operator-boolean kondisi_2;
```

Contoh :

Menampilkan kolom **nip**, **nama** , **sex** dan **kd_golongan** dari tabel **pegawai**, hanya jika pada kolom **sex** bernilai = '**L**' atau **kd_golongan='02'**

```
SELECT nip,nama,sex,kd_golongan FROM pegawai  
WHERE sex='L' OR kd_golongan='02';
```

- a. Percobaan :
Tampilkan semua kolom dan baris pada tabel pegawai jika kolom **sex='P'** dan **kd_golongan = '01'**.
Amati dan catat hasil percobaan tersebut.
Buat analisis dari hasil tersebut.

Percobaan-1

6. Memilih Baris/Record Tertentu menggunakan operator NOT

```
SELECT nama_kolom FROM nama_tabel  
WHERE NOT kondisi;
```

Contoh :

Menampilkan kolom **nama** , **sex** dan **tempat_lahir** dari tabel **pegawai**, hanya jika pada kolom **sex** tidak bernilai = '**L**'

```
SELECT nama,sex,tempat_lahir FROM pegawai  
WHERE NOT (sex='L');
```

- a. Percobaan :
Tampilkan semua kolom dan baris pada tabel pegawai jika kolom sex tidak bernilai 'P' atau nip tidak sama dengan '132310742'
Amati dan catat hasil percobaan tersebut.
Buat analisis dari hasil tersebut.

Percobaan-1

7. Memilih Baris/Record Tertentu menggunakan operator BETWEEN

```
SELECT nama_kolom FROM nama_tabel  
WHERE nama_kolom BETWEEN nilai_1 AND nilai_2;
```

Contoh :

Menampilkan kolom **nama** dan **tgl_lahir** dari tabel **pegawai**, hanya jika pada kolom **tgl_lahir** bernilai antara '01/01/1980' dan '30/12/1990'

```
SELECT nama,tgl_lahir FROM pegawai  
WHERE tgl_lahir BETWEEN '01/01/1980' AND '30/12/1990';
```

- a. Percobaan :
Tampilkan semua kolom dan baris pada tabel pegawai jika kolom tgl_lahir tidak bernilai antara '01/01/1990' dan '01/01/1995'.
Amati dan catat hasil percobaan tersebut.
Buat analisis dari hasil tersebut.

Percobaan-1

8. Memilih Baris Tertentu menggunakan operator NOT BETWEEN

```
SELECT nama_kolom FROM nama_tabel  
WHERE NOT nama_kolom BETWEEN nilai_1 AND nilai_2;
```

Contoh :

Menampilkan kolom **nama** dari tabel **pegawai**, hanya jika pada kolom **nama** tidak bernilai antara 'E' dan 'K'

```
SELECT nama FROM pegawai  
WHERE NOT nama BETWEEN 'E' AND 'K';
```

- a. Percobaan :
Tampilkan kolom nama pada tabel pegawai jika kolom nama bernilai tidak bernilai antara 'C' dan 'K'
Amati dan catat hasil percobaan tersebut.
Buat analisis dari hasil tersebut.

Percobaan-1

9. Memilih Baris Tertentu menggunakan operator IN dan NOT IN

Operator IN berguna untuk melakukan pencocokan dengan salah satu yang ada pada suatu daftar nilai. Sebagai contoh :

```
SELECT nama FROM pegawai WHERE nip = '132310742'  
OR nip = '132310743' OR nip = '132310744';
```

Keterangan :

Perintah diatas digunakan untuk memperoleh nama-nama pegawai yang sesuai dengan nilai nip yang disebutkan.

Perintah diatas dapat ditulis dengan operator IN , yaitu sebagai berikut :

```
SELECT nama FROM pegawai  
WHERE nip IN ('132310742' , '132310743' , '132310744');
```

- a. Percobaan :
Jalankan perintah diatas dan amati hasilnya.
Gunakanlah operator NOT IN untuk kondisi diatas.
Buat analisis dari hasil tersebut.

Percobaan-1

10. Memilih Baris Tertentu menggunakan operator LIKE dan NOT LIKE

Operator LIKE berguna untuk mencari nilai data dalam suatu kolom yang nilai datanya mendekati/menyerupai dengan nilai yang di bandingkan.

```
SELECT nama_kolom FROM nama_tabel  
WHERE nama_kolom LIKE 'nilai';
```

Nilai data dapat diberi atau ditambahkan tanda wildcard berupa garis bawah (_) atau persen (%).

a. Contoh penggunaan Wildcard :

%a% → cocok dengan apa saja yang mengandung karakter 'a' atau 'A'

%a → cocok dengan apa saja yang berakhiran huruf a atau A. (VARCHAR)

a% → cocok dengan apa saja yang berawalan huruf 'a' atau 'A'.

a. Percobaan :

Tampilkan kolom nama jika nilai dalam kolom nama diawali dengan huruf 'e' atau 'E'

Tampilkan kolom nama jika nilai pada kolom nama tidak diawali dengan huruf 'e' atau 'E'

Amati, catat dan buatlah analisis dari hasil tersebut.

Percobaan-1

11. Memilih Baris Tertentu dgn operator NULL dan NOT NULL

Operator **NULL** berguna untuk mencari data dalam suatu kolom dan baris yang bernilai null.(belum ada/tidak ada data) dan **NOT NULL** (sebaliknya).

```
SELECT nama_kolom FROM nama_tabel  
WHERE nama_kolom IS NULL;
```

```
SELECT nama_kolom FROM nama_tabel  
WHERE nama_kolom IS NOT NULL;
```

- a. Percobaan :
- Tampilkan kolom nama jika nilai dalam kolom sex bernilai null.
 - Tampilkan kolom nama jika nilai dalam kolom sex tidak bernilai null.
 - Amati, catat dan buatlah analisis dari hasil tersebut.

Klausu

Pada penggunaan pernyataan SELECT, klausa-klausa yang dapat terlibat :

- ORDER BY
- GROUP BY
- HAVING
- LIMIT

Percobaan-2

1. Mengurutkan data menggunakan **ORDER BY**

Klausa **ORDER BY** digunakan untuk mengurutkan data hasil query. Pengurutan dapat Dilakukan baik secara ascending maupun descending baik berdasarkan sebuah kolom Atau lebih.

a. Mengurutkan data berdasarkan satu kolom (urut naik) - ASC

```
SELECT * FROM nama_tabel  
ORDER BY nama_kolom;
```

Percobaan :

Tampilkan semua data pada tabel pegawai dengan kolom nama harus urut naik (ascending).

b. Mengurutkan data berdasarkan beberapa kolom dan urut turun (descending). -DESC

```
SELECT * FROM nama_tabel  
ORDER BY nama_kolom_1, nama_kolom_2 DESC;
```

Percobaan :

Tampilkan semua data pada tabel pegawai dengan kolom nama harus urut turun (descending).

Percobaan-2

c. Mengurutkan data berdasarkan nomor kolom

```
SELECT nama_kolom_1, nama_kolom_2 FROM nama_tabel  
ORDER BY nomor_kolom jenis_urutan;
```

Contoh :

Menampilkan nilai data pada kolom nip dan nama pada kolom kedua (nama) harusurut naik.

```
SELECT nip, nama FROM nama_tabel  
ORDER BY 2 DESC;
```

Percobaan :

Tampilkan semua data pada tabel pegawai dengan kolom 4 harusurut turun (descending).

Percobaan-2

2. Pengelompokkan data menggunakan **GROUP BY**

```
SELECT nama_kolom FROM nama_tabel  
GROUP BY nama_kolom;
```

Contoh :

Pengelompokan nilai data pada kolom tempat_lahir dari tabel pegawai.

```
SELECT tempat_lahir FROM pegawai  
GROUP BY tempat_lahir;
```

Percobaan :

- a. Tampilkan pengelompokkan nilai data kolom sex pada tabel pegawai.
- b. Amati dan catat hasilnya.

Percobaan-2

3. Mengenal klausa **HAVING**

Pemakaian klausa **HAVING** terkait dengan klausa **GROUP BY**. Kegunaanya adalah untuk Menentukan kondisi bagi **GROUP BY**. Kelompok yang memenuhi **HAVING** saja Yang akan ditampilkan.

```
SELECT nama_kolom FROM nama_tabel  
GROUP BY nama_kolom HAVING kondisi;
```

- a. Percobaan :
Tampilkan pengelompokkan data nilai kolom tempat_lahir jika nilai kolom tempat_lahir tidan sama dengan 'Lampung'.
Amati, catat dan buatlah analisis dari hasil tersebut.

Percobaan-2

4. Mengenal klausa **LIMIT**

Klausa **LIMIT** digunakan untuk membatasi jumlah baris/record yang dihasilkan Oleh perintah query.

```
SELECT nama_kolom FROM nama_tabel  
LIMIT nilai;
```

- a. Percobaan :
Tampilkan semua kolom pada tabel pegawai sebanyak record 4 pegawai.
Amati, catat dan buatlah analisis dari hasil tersebut.

Fungsi Agregat

Pada penggunaan pernyataan SELECT, Fungsi_Fungsi Agregat yang dapat terlibat :

- AVG → memperoleh nilai rata-rata
- COUNT → memperoleh nilai cacah data
- MAX → memperoleh nilai terbesar
- MIN → memperoleh nilai terkecil
- SUM → memperoleh nilai jumlah

Percobaan-3

1. Fungsi **AVG()**

```
SELECT AVG(nama_kolom) FROM nama_tabel;
```

Contoh :

Menampilkan nilai rata-rata berdasarkan kolom gapok dari tabel golongan.

```
SELECT AVG(gapok) FROM golongan;
```

Percobaan :

- a. Jalankan perintah diatas.
- b. Amati dan catat hasilnya.

Percobaan-3

2. Fungsi **COUNT()**

```
SELECT COUNT(nama_kolom) FROM nama_tabel;
```

Contoh :

Menampilkan nilai total berdasarkan kolom nama dari tabel pegawai.

```
SELECT COUNT(nama) FROM pegawai;
```

Percobaan :

- a. Jalankan perintah diatas.
- b. Amati dan catat hasilnya.
- c. Jalankan perintah berikut :
SELECT sex, count(*) FROM pegawai GROUP BY sex;
- d. Amati, catat dan buatlah analisis hasil querynya.

Percobaan-3

3. Fungsi **MAX()** dan **MIN()**

```
SELECT MAX(nama_kolom) FROM nama_tabel;
```

```
SELECT MIN(nama_kolom) FROM nama_tabel;
```

Contoh 1 :

Menampilkan nilai terbesar berdasarkan kolom gapok dari tabel golongan.

```
SELECT MAX(gapok) FROM golongan;
```

Contoh 2 :

Menampilkan nilai terkecil berdasarkan kolom gapok dari tabel golongan.

```
SELECT MIN(gapok) FROM golongan;
```

Percobaan-3

3. Fungsi **SUM()**

```
SELECT SUM(nama_kolom) FROM nama_tabel;
```

Contoh :

Menampilkan jumlah nilai berdasarkan kolom gapok dari tabel golongan.

```
SELECT SUM(gapok) FROM golongan;
```