Data Manipulation Language (DML)

a. SELECT

SINTAK :
SELECT [**DISTINCT**] daftar_select | *
INTO nama_tabel_baru
FROM daftar_tabel
WHERE kondisi_pencarian
[**AND**,**NOT**,**OR**] [**IS NOT NULL**] [**IN**,**NOT IN**] [**BETWEEN**] [**LIKE**]
ORDER BY daftar_order [**ASC** / **DESC**]
GROUP BY daftar_group_by
HAVING kondisi_pencarian

Contoh Penerapan :

```
- use Northwind
- SELECT * FROM customers
- SELECT customerid,companyname,contactname
  FROM customers
```

DISTINCT

Jika statemen SELECT tidak menyertakan column dengan constraint Primary Key, maka kemungkinan tabel yang ditampilkan berisi data yang sama (duplikat). Dengan menggunakan statement DISTINCT kita bisa mengeliminasi data/record yang sama.

Contoh :

```
- INSERT INTO Mahasiswa VALUES('1200001', 'Amin','Jl. PHH Mustofa 60', 'L')
  Go
  INSERT INTO Mahasiswa VALUES('1200001', 'Amin','Jl. PHH Mustofa 60', 'L')
  Go
  SELECT * FROM Mahasiswa

- SELECT DISTINCT Nim,Nama,Alamat FROM Mahasiswa
```

INTO

Kita dapat menggunakan statement INTO untuk membuat tabel baru sekaligus dengan isinya.
Misal kita akan membuat tabel baru yang berisi FirstName dan Lastname dari tabel Employees, data tersebut dimasukan ke dalam tabel NamaEmployees :

```
SELECT FirstName,LastName
INTO NamaEmployees
FROM Employees
```

Hasilnya adalah bukan tampilan data, tetapi hanya pesan bahwa telah dibuatkan tabel baru dengan nama NamaEmployes.
Untuk menampilkannya kita harus memberikan perintah SELECT pada tabel baru tersebut : `SELECT * FROM NamaEmployees`

FROM

Statemen FROM diperlukan dalam setiap perintah SELECT. Statemen ini menyatakan tabel yang merupakan asal data.

WHERE

Dengan menggunakan statemen WHERE, maka seleksi dilakukan tidak pada seluruh record/data, melainkan hanya pada record yang memenuhi syarat.
SQL-Server mengenal beberapa operator :

Operator	Keterangan
=	sama dengan
< >	tidak sama dengan
<	lebih kecil dari
< =	lebih kecil atau sama dengan
>	lebih besar dari
> =	lebih besar atau sama dengan

Contoh :

```
- SELECT CustomerID,CompanyName,ContactName
  FROM Customers
  WHERE ContactTitle = 'Marketing Manager'
- SELECT * FROM "Order Details"
  WHERE Discount < 0.10
```

AND, **NOT** dan **OR**

Untuk menggabungkan dua atau lebih kondisi, SQL-Server menggunakan logical AND dan OR, untuk menyangkal (negasi) dua atau lebih kondisi digunakan logical NOT. Tabel berikut memperlihatkan bagaimana SQL-Server melakukan interprestasi jika menghubungkan dua kondisi (p dan q) :

p	q	NOT p	p AND q	p OR q
T	T	F	T	T
T	F	F	F	T
T	Unknown	F	Unknown	T
F	T	T	F	T
F	F	T	F	F
F	Unknown	T	F	Unknown
Unknown	T	Unknown	Unknown	T
Unknown	F	Unknown	F	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown

Contoh :

```
SELECT CustomerID,CompanyName,City
FROM Customers
WHERE ContactTitle='Owner'
AND Country='USA'
```

IS NOT NULL

NULL adalah nilai yang belum diisi, artinya jika dalam suatu pengisian data salah satu field tidak diisi, maka nilai dari field tersebut adalah NULL.
NULL bukan berarti spasi atau nol (0).

Contoh : `SELECT CustomerID,Region FROM Customers WHERE Region IS NOT NULL`

IN dan **NOT IN**

Dengan kondisi IN kita bisa membatasi baris data yang ingin kita tampilkan berdasarkan suatu kelompok nilai tertentu.
Kondisi NOT IN adalah kebalikan dari kondisi IN.

Contoh :

```
- SELECT CustomerID,CompanyName,ContactName,City
  FROM Customers
  WHERE City IN('London','Paris')
- SELECT CustomerID,CompanyName,ContactName,City
  FROM Customers
  WHERE City NOT IN('London','Paris')
```

BETWEEN

BETWEEN menyederhanakan pencarian 'antara' (range).
Contoh :

```
 SELECT OrderID,UnitPrice,Discount
 FROM "Order Details"
 WHERE Discount BETWEEN 0.1 AND 0.25
```

LIKE

Umumnya LIKE digunakan untuk permintaan yang mencari suatu teks berdasarkan kata depan (prefix), kata tengah dan kata akhir (sufix).
Contoh :

```
- SELECT CustomerID,CompanyName,ContactName,City
  FROM Customers
  WHERE City LIKE 'Po%'
- SELECT CustomerID,CompanyName,ContactName,City
  FROM Customers
  WHERE City LIKE '%an%'
- SELECT CustomerID,CompanyName,ContactName,City
  FROM Customers
  WHERE City LIKE '%i'
```

ORDER BY

Tampilan dari hasil SELECT dapat disusun (sort) menurut satu atau beberapa kolom. Kita bisa menggunakan staemen ORDER BY. Susunan dapat diatur secara Menurun (ASCending) atau Menaik (DESCending).

Contoh :

```
- SELECT CustomerID,CompanyName,ContactName
  FROM Customers
  ORDER BY CustomerID    --ASC boleh tidak ditulis
- SELECT CustomerID,CompanyName,ContactName
  FROM Customers
  ORDER BY CompanyName DESC
```

Fungsi Aggregate

SQL-Server juga dapat memiliki beberapa fungsi aggregate yang dapat digunakan untuk mendapatkan suatu nilai yang merupakan hasil dari perhitungan sekelompok baris data.

Fungsi	Keterangan
AVG(ekspresi)	Nilai rata-rata dari column yang tidak null
COUNT(*)	Menghitung jumlah baris data dalam tabel
COUNT(ekspresi)	Menghitung jumlah baris data dalam tabel yang tidak null
COUNT(distinct ekspresi)	Count hanya menghitung column yang unik
MAX(ekspresi)	Mencari nilai tertinggi
MIN(ekspresi)	Mencari nilai terkecil
SUM(ekspresi)	Menjumlahkan baris nilai data dalam tabel yang bukan null

STDEV(ekspresi)	Meghitung standar deviasi dari nilai column yang bukan null
STDEVP(ekspresi)	Meghitung standar deviasi dari suatu populasi column yang bukan null
VAR(ekspresi)	Meghitung varians dari nilai column yang bukan null
STDEVP(ekspresi)	Meghitung varians dari suatu populasi column yang bukan null

Contoh :

```
- SELECT AVG(UnitPrice) AS ‘Rata-rata’
  FROM “Order Details”
- SELECT SUM(UnitPrice) AS ‘Jumlah’, AVG(UnitPrice) AS ‘Rata’
  FROM “Order Details”
- SELECT COUNT(DISTINCT City) AS ‘Jumlah Record’
  FROM Customers
- SELECT COUNT(*) AS ‘Jumlah Record’
  FROM Customers
```

GROUP BY

GROUP BY digunakan untuk menyeleksi baris data kedalam kelompok-kelompok baris data yang memiliki nilai sama. GROUP BY membuat himpunan nilai sebelum dihitung fungsi-fungsi aggregate.

Contoh :

```
SELECT City,COUNT(*) AS 'Jumlah Customer'
FROM Customers
GROUP BY City
```

HAVING

Kita dapat menggunakan statemen HAVING untuk memberi batasan terhadap baris-baris data yang dihasilkan oleh tugas aggregate.

Contoh :

```
SELECT City,COUNT(*) AS 'Jumlah Customer'
FROM Customers
GROUP BY City
HAVING COUNT(*)>1
```

Join

Operator join dipakai untuk mencari data dari beberapa tabel berdasarkan hubungan logis tabel-tabel tersebut. Join menyatakan cara SQL Server memakai data dari sebuah tabeluntuk memilih data dari tabel lain. SQL Server dapat menggabungkan tabel-tabel sampai 256 tabel.

Ada beberapa macam penggunaan join, antara lain INNER JOIN, OUTER JOIN, dan SELF JOIN.

INNER JOIN

Inner join merupakan suatu operator perbandingan untuk mencari suatu data/baris record dari dua tabel atau lebih berdasarkan pada nilai-nilai yang terdapat pada kolom dari masing-masing tabel.

Contoh :

```
use pubs
select t.title,p.pub_name
from publishers p inner join titles t
on p.pub_id=t.pub_id
order by t.title
```

Contoh diatas dapat diubah memakai inner join style baru yang memakai kondisi where.

```
select t.title,p.pub_name
from publishers p,titles t
where p.pub_id=t.pub_id
order by t.title
```

Juga dapat mengambil data dari beberapa tabel :

```
select t.title,p.pub_name,i.au_id
from titles t
inner join publishers p on t.pub_id=p.pub_id
inner join titleauthor i on t.title_id=i.title_id
```

atau

```
select t.title,p.pub_name,i.au_id
from titles t,publishers p,titleauthor i
where t.pub_id=p.pub_id
and t.title_id=i.title_id
```

OUTER JOIN

Apa gunanya outer join ? Kegunaan utamanya adalah untuk mencari record-record piatu, artinya record yang ada di sebuah tabel, tetapi tidak ada pasangannya di tabel lain.

Outer join akan menghasilkan semua data dari sebuah tabel dan membatasi data dari tabel lainnya. SQL Server mempunyai tiga tipeouter jin yaitu : **left**, **right**, dan **full**. Semua baris dari tabel sebelah kiri diacu dengan sebuah left outer join, semua baris dari tabel sebelah kanan diacu dengan sebuah right outer join, dan semua baris dari kedua tabel dihasilkan oleh sebuah full outer join.

Contoh :

```
use pubs
-- LEFT OUTER JOIN
select a.city,p.city,a.au_fname,p.pub_name
from authors a left outer join publishers p
on a.city=p.city

-- RIGHT OUTER JOIN
select a.city,p.city,a.au_fname,p.pub_name
from authors a right outer join publishers p
on a.city=p.city

-- FULL OUTER JOIN
select a.city,p.city,a.au_fname,p.pub_name
from authors a full outer join publishers p
on a.city=p.city
```

SELF JOIN

Sesuai dengan namanya, self join menghubungkan baris-baris dari sebuah tabel dengan baris-baris lainnya dalam tabel yang sama.

Misalnya anda akan menampilkan data semua author yang tinggal di kota dan kodepos yang sama. Anda harus membandingkan tabel author dengan tabel author.

```
select a1.au_fname,a2.au_fname,a1.city,a1.zip
from authors a1,authors a2
where a1.city=a2.city and a1.zip=a2.zip
and a1.au_fname<a2.au_fname
order by a1.city
```

Subquery

Subquery adalah sebuah pernyataan SELECT didalam sebuah pernyataan SELECT yang lain.

Contoh :

```
use Northwind
select productname from products
where unitprice=
  ( select unitprice from products
    where productname='Outback Lager' )
select distinct companyname,country
from customers
where postalcode in
  ( select postalcode from suppliers )
```

UNION

Kita dapat menggabungkan hasil beberapa query dengan operator UNION. Defaultnya, baris-baris duplikasi akan dihilangkan. Untuk mempertahankan duplikasi dipakai kata kunci ALL.

Misal kita akan menampilkan semua Customer dan Supplier dari negara USA, diurutkan brdasarkan CompanyName :

```
use Northwind
select companyname,contactname
from suppliers
where country='USA'
union
select companyname,contactname
from customers
where country='USA'
order by companyname
```

b. INSERT

Kita dapat menggunakan statement INSERT untuk memasukan data (record) kedalam tabel.

SINTAK :
INSERT INTO *table_name*
    VALUES *(value_1,value_2,.....,value_n)*
INSERT INTO *table_name(column_1, column_2,... ,column_n)*
    VALUES *(value_1,value_2,.....,value_n)*

Contoh Penerapan :

```
create database mhs
go
use mhs
go
create table mahasiswa
( nim varchar(7) not null primary key,
  nama varchar(25) not null unique,
  alamat varchar(30)not null,
  no_telp varchar(12),
  tempat_lhr    varchar(15),
  tanggal_lhr datetime,
  agama   varchar(15),
  gol_darah varchar(2)
)
go
insert into mahasiswa
values('1200001','Asep','Jl. Phh Mustopa 11',
       '0227070707','Bandung','1982-12-29','islam','A')
insert into mahasiswa
values('1200002','Budi','Jl. Phh Mustopa 21',
       '0227070708','Bandung','1982-12-30','islam','B')
insert into mahasiswa
values('1200003','Cecep','Jl. Phh Mustopa 31',
       '0227070709','Bandung','1982-12-31','islam','AB')

insert into mahasiswa(nim,nama,alamat)
values('1200004','Dedi','Jl. Phh Mustopa 41')

select * from mahasiswa
```

c. UPDATE

Kita dapat menggunakan statement UPDATE untuk memodifikasi data (record) di dalam tabel.

SINTAK :

UPDATE *table_name*
SET *column_name* **=** *new_value*
WHERE *column_name* **=** *value*

Contoh Penerapan :

```
use mahasiswa
go
update mahasiswa
set nama='Abdul Rohim'
where nim='1200001'

update mahasiswa
set nama='Budi Haryono',
    alamat='Jl. Supratman 82'
where nim='1200002'

update mahasiswa
set nama='Asep Sunandar Sunarya',
     no_telp='08567070707',
     alamat='Jl. Jakarta 82'
where nama='Cecep'

update mahasiswa
set no_telp='081380808080',
     gol_darah='O',
```

```
    tempat_lhr='Jakarta',
    tanggal_lhr='1983-11-02'
where nim='1200004'

select * from mahasiswa
```

d. DELETE

Kita dapat menggunakan statement DELETE untuk menghapus data (record) di dalam tabel.

SINTAK :
DELETE *table_name*
WHERE *column_name* **=** *value*

Contoh Penerapan :

```
use mahasiswa
go
delete mahasiswa
where nim='1200001'

delete mahasiswa
where nama='Budi Haryono'

select * from mahasiswa
```