BAB 10

SQL (Structured Query Language)

RDBMS adalah proram komputer yang dirancangan untuk pengelolaan data dengan melakukan penyimpanan, pembaruan dan pengambilan data. SQL adalah bahasa khusus yang digunakan untuk mengakses dan mengelola RDBMS. Bahasa SQL telah distandarkan, namun demikian telah berkembang banyak varian dan bentuk sesuai dengan kebutuhan vendor RDBMS.

SQL adalah bahasa yang mulanya berorientasi pada basia data relasional. Bahasa ini menghilangkan banyak pekerjaan yang perlu dilakukan pemrogram / pengembang berkaitan dengan operasi – operasi terhadap data bgila dibanding denan menggunakan bahasa general purpose. SQL adalah bahasa yang menggabungkan fitur – fitur bahasa query formal aljabar relasional dan bahasa *query* formal kalkulus relasional. Meski SQL diacu sebagai bahasa *query* (yaitu bahasa untuk meminta data) namun sesungguhnya SQL bukan hanya sekedar bahasa query terhadap basis data. SQL juga berisi fasilitas untuk mendifinisikan struktur data, modifikasi struktur data, serta digunakan menspesifikasikan *constraint – constraint* integritas dan keamanan data.

10.1. Sejarah SQL

SQL diawali publikasi makalah E.F. Codd (1970) mengenai model relasional *:"A Relational Model of Data for Large Shared Data Banks*". Pada tahun 1974, D.Chamberlin an R.F. Boyce mengembangkan bahasa *query* untuk memanipulasi dan mengekstrak data dari basis data relasional, dan definisi dengan *"Structured English Query Language"* yang disingkat sebagai *SEQUEL*, yang dikemukakan dalam makalah berjudul *"SEQUEL = A Structured English Query Language"*. Kemudian *SEQUEL* berevolusi menjadi versi revisi yaitu *SEQUEL/2* pada tahun 1976. Orang mengejanya dengan SQL dan menyebutkan dengan "si-quel" meski ejaan resminya adalah "s-q-l".

Bahasa SQL mempunyai beberapa bagian yaitu:

1. Bahasa pendifinisian data (DDL-*Data Definition Language) untuk pendifinisian skema relasi, penghapusan relasi dan memodifikasi skema relasi*.
2. Bahasa manipulasi data interaktif (DML-*Data Manipulation Language*), berisi bahasa *query* berbasis aljabar relasional dan kalkulus relasional tupel, memasukkan tupel, menghapus tupel dan melakukan modifikasi tupel.
3. Pendifinisian *View* untuk mendifiniskan *View.*
4. Kendali transaksi untuk menspesifikasikan permulaan dan akhir transaksi.

5. *Embedded SQL* dan *dynamic SQL* yang mendifiniskan cara kalimat SQL ditempelkan dibahasa pemrograman umum seperti C, C++, Java, PL/1, Cobol, Pascal dan Fortran.
6. Integritas, bagian dari DDL untuk menspesifikasikan konstrain – konstrain integritas dimana data disimpan yang harus dipenuhi basis data. Pembaruan yang melanggar konstrain – kontrain integritas ditolak.
7. Otoriasi, bagian DDL uang menspesifikasikan hak – hak akses terhadap relasi dan view.

Bahasa basis data harus memungkinkan pemakai melakukan hal – hal sebagai berikut:

1. mencipakan baisis data dan struktur – struktur relasi.
2. Melakukan manajemen data tingkat dasar seperti penyisipan (*insertion*), modifikasi (*modification*) struktur dan data, serta penghapusan (*deletion*).
3. Membentuk *query* sederhana dan kompleks yang mentransformasi data di basis data menjadi informasi.
4. melakukan tugas – tugas dengan seminimal mungkin memakai struktur dan sintaks perintah relatif mudah dipelajari.
5. harus portabel, yaitu memenuhi suatu standard sehingga dapat menggunakan struktur dan sintaks perintah beragamam DBMS lain.

10.2. Subdivisi SQL

SQL (*Structured Query Language*) dapat dikelompokkan menjadi 3 (tiga), yaitu DDL (*Data Definition Language*), DML (*Data Manipulation Language*) dan DCL (*Data Control Language*).

a. DDL (*Data Definition Language*)

DDL disebut sebagai bahasa untuk pendefinisian skema (Schema Definition Language) yag berisi perintah – perintah untuk menciptakan objek – objek basis data (table, indeks, view dan lainnya).

b. DML (*Data Manipulation Language*)

DML adalah sekelompok perintah yang menentukan dan melakukan manipulasi nilai – nilai didalam suatu table pada suatu waktu yang diinginkan.

c. DCL (*Data Control Language*)

DCL berisi fitur – fit

ur yang menentukan aksi yang dapat dilakukan pemakai terhadap objek basis data seperti basisdata, tale, view dan lainnya. Pada ISO, DCL termasuk sebagai bagian dari DDL, selain itu dapat ditambahkan bagian berikut:

- View definiton, SQL DDL untuk perintah mendifinisikan *View.*

- *Transaction control*, untuk menspesifikasikan awal an akhir transaksi dan melakukan pengendalian transaksi.
- *Embedded SQL* dan *dynamic SQL*, mendifiniskan cara kalimat SQL dapat ditempelkan dibahasa pemrograman umum seperti C, C++, Java, PL/1, Cobol, Pascal, Fortran dan sebagainya.
- *Integrity*, perintah untuk menspesifikasikan konstrain – konstrain integritas dimana data disimpan di basisdata yang harus dipenuhi DBMS. Pembaruan yang melanggar konstrain – kontrain integritas ditolak.

10.3. SQL – DDL

DDL (*Data Definition Language*) memungkinkan kita membuat dan menghancurkan objek – objek basis data (database/schema, domain, table, view, dan index). DBMS akan menggunakan informasi deskripsi struktur basis data saat menterjemahkan kalimat DML menjadi perintah – perintah ke manajer basis data. Informasi ini diperoleh dari data dictionary / directory. Setiap kalimat terdapat kalimat DDL yang baru maka terdapat perubahan pada data *dictionary / directory*. *System catalog* akan secara otomatis dibuat pada saat pembuatan basis data, serta kemudian diperbaharui begitu terdapat eksekusi kalimat DDL.

DDL berbeda untuk dialek – dialek SQL yang berbeda, dibawah ini kita akan menggunakan ISO SQL sebagai acuan. Kalimat DDL mendefiniskan struktur data dengan menciptakan dan mengelola basis data dan objek – objek basis data seperti *table* dan *store procedure*. Kebanyakan kalimat DDL mempunyai bentuk sebagai berikut

- *CREATE* object_name
- *ALTER* object_name
- *DROP* object_name

Secara *default,* hanya anggota administrator yang dapat mengeksekusi kalimat DDL. Jika pemakai – pemakai berbeda menciptakan objek-objeknya sendirin di basidata maka masing – masing pemilik objek perlu memberikan wewenang yang cocok untuk masing – masing pemakai objek. Keperluan pemberian wewenang ini menyebabkan halangan administratif sehingga kita seharusnya menghindari pemakai – pemakai berbeda menciptakan objeknya sendiri.

10.3.1 Identifier SQL

Identifier SQL digunakan untuk mengidentifikasi objek – objek di basis data seperti nama table, view dan kolom. Karakter – karakter yang dapat digunakan harus terdapat pada himpunan

karakter (character set). Standard ISO menyediakan himpunan karakter default terdiri – dari karakter huruf kapital (A .. Z), huruf kecil (a .. z) dan karakter garis bawah (_). Berikut ini adalah batasan – bataan penamaan untuk identifier SQL, yaitu:

- Identifier tidak boleh lebih panjang dari 148 karakter (kebanykaan dialek lebih pendek).
- Identifier harus dimulai dengan huruf.
- Identifier tidak boleh berisi spasi.

10.3.2. Tipe data ISO SQL

Pada saat menciptkan tabel, kita harus mendifiniskan tipe data dari msing – masing kolom (field) pada tabel tersebut. Tipe data menspesfikasikan tipe informasi (karakter, angka atau tanggal) yang dapat ditangani kolom termasuk cara data tersebut disimpan.

a. ISO SQL

Ada 6 (enam) tipe data dalam standard ISO, yaitu:

No	Tipe Data	Deklarasi		
1.	Karakter	CHAR	VARCHAR	
2.	Bit	BIT	BIT VARYING	
3.	Numerik eksak	NUMERIC	DECIMAL	INTEGER SMALLINT
4.	Numerik riil	FLOAT	REAL	DOUBLE PRICISION
5.	Waktu tanggal	DATE	TIME	
6.	interval	INTERVAL		

Tabel 10.1. Tipe data ISO

b. MS SQL Server

MS SQL Server mendukung beragam tipe – tipe data sistem. SQL Server juga memungkinkan tipe data yang didefinisikan pemakai yang dibangun berdasarkan tipe – tipe data yang disediakan sistem.

Pada persoalan DDL ini, meskipun hampir seluruh RDBMS mengaku memenuhi standard misalnya ANSI SQL, maka perluasan yang dilakukan dari masing – masing RDBMS itu menjadikan RDBMS – RDBMS tersebut menjadi tidak kompatible satu dengan lainnya.

Saran praktis alam melakukan pendefinisian basis data adalah sebaiknya kita melakukan seluruh proses ini menggunakan tools otomatis seperti ER-WIN atau Power Designer dari Powersoft Inc. dan sebagainya. Dengan menggunakan tools yang netral terhadap vendor ini, kita dapat membangkitkan beraneka ragam sql script yang dapat

dieksekusi pada beraneka ragam RDBMS dengan hanya mendefinisikan taarget RDBMS yang dikehendaki.

b.1. Didefiniskan Sistem

Pada MS SQL Server menyediakan berbagai tipe data yang berbeda, tipe – tipe data tertentu mempunyai beberapa tipe dengan asosiasinya denan tipe – tipe data yang diberikan *SQL Server*. Contohnya kita dapat menggunakan tipe data int, decimal atau float untuk menyiman data numeric.

Tabel berikut memetakan tipe data yang umum ke tipe data yang didukung SQL Server, tabel juga berisi sinonim tipe data untuk kompatibilitas ANSI.

Tipe data	Tipe data disediakan system	Sinonim dengan ANSI	Jumlah byte
Binary	Binary[(n]) Varbinary[(n])	- binary varying[(n])	1-8000
Character	Char[(n)] Varchar[[n)]	Character[(n)] char[acter]varying[(n)]	1-8000 (8000 characters)
Unicode character	Nchar[(n)] Nvarchar[(n)]	National char[acter][(n)] National char[acter]varying[(n)]	2-8000 (1-4000 characters)
Date and time	Datetime, smalldatetime	-	8 (24 byte integers) 4 (22 byte integers)
Exact numeric	Decimal[(p[,s])] numeric[(p[,s])]	dec	5-17
Approximate numeric	Float[(n)] Real	Double precision or Float[(n)]	4-8 4
Global identifier	Uniqueidentifier	-	16
Integer	Int smallint, tinyint	Integer -	4 2,1
Monetary	Money, smallmoney	-	8,4
Special	Bit, cursor,sysname, timestamp	-	1, 0-8
Text and image	Text, image	-	0-2 GB
Unicode text	Ntext	National text	0-2 GB

Tabel 10.2. Tipe data SQL Server

b.2. Didefiniskan Pemakai

Tipe data yang didefiniskan pemakai (user-defined data type) berdasarkan tipe data yang disediakan sistem. Kita dimungkinkan mendefiniskan sendiri tipe data untuk menjamin konsistensi saat bekerja pada table – tabel atau basisdata – basisdata berbeda. Tipe data didefinisikan pemakai tidak mengijikan pendifinisian struktur atau tipe data kompleks.

SQL server menyediakan stored procedure sebagai berikut:

1. sp_addtype, untuk menciptakan tipe data yag didefinisikan pemakai.
2. sp_droptype, untuk menghapus tipe data yag didefinisikan pemakai.

b.2.1. Penciptaan tipe data

Sintaksnya adalah:

Sp_addtype type, system_data_type [,'NULL,I,NOT NULL']

Berikut ini adalah contoh 3 (tiga) tipe data yang didefiniskan oleh pemakai melalu store procedure.

```
EXEC sp_addtype isbn,'smallint',NOT NULL
EXEC sp_addtype zipcode,'char(10)',NULL
EXEC sp_addtype alamat,'varchar(90)',NULL
```

Terdapat beberapa petunjuk penciptaan tipe data oleh pemakai dan menyeimbangkan ukuran penyimpanan:

1. Jika panjang kolom beragam, gunakan salah satu dari tipe data variable, misalkan kolom alamat kita gunakan varchar bukan *char(fixed)* untuk menghemat penggunaan ruang penyimpanan.
2. Tipe data *integer, date* dan *time* dan *monetary* mendukung jangkauan berbeda berdasarkan ukuran penyimpanan. Jika kita menggunakan tipe data *tinyint* untuk menyimpan identifier dalam basisdata, maka kita akan bermasalah saat memutuskan penyimpanan data angka 256.
3. Tipe data *numeric*, ukuran dan level presisi yang diperlukan membantu menentukan piihan, umumnya kita menggunakan *decimal.*
4. Jika penyimpanan lebih dari 8000 *byte*, gunakan *text* atau *image*, jika kurang dari 8000 gunakan *char* , *varchar*, atau *binary.*

b.2.2. Penghilangan tipe data

Untuk menghilangkan tipe data yang telah dibuat dengan prosedur sebelumnya, unutk melakukan penghapusan, sintaksnya adalah:

```
Sp_droptype type
```

Berikut ini adalah contoh untuk melakukan drop untuk tipe data isbn yang telah didefinisikan sebelumnya:

```
Sp_droptype isbn
```

10.3.3. Pendifinisian Basis Data

Dalam pendefinisan basis data ini meliputi pembuatan basis data, pembuatan *table*, *indeks* dan aturan penamaan.

a. Pembuatan *database*

Proses pembuatan basis data sangat berbeda antara DBMS satu dengan lainnya. Pada sistem *multiuser*, otoritas penciptaan basis data biasanya berada pada DBMS (Database Adminstrator). Standard ISO tidak menspesifikasikan cara penciptaan basis data dan masing – masing dialek SQL mempunyai pendekatan berbeda. Teknik yang digunakan *INGRES* dan *ORACLE* atau *SQL Server* adalah sebagai berikut:

- INGRES menyediakan program utilitas khusus untuk meng *CREATEDB*, yaitu *DESTROYDB.*
- *ORACLE* dan *SQL SERVER* menciptakan basis data sebagai bagian proses instalasi. Untuk kebanyakan bagian, tabel – tabel pemakai ditempatkan pada bais data tunggal, pembuatan database harus unik dalam suatu *server database.*

Menurut standard ISO, relasi – relasi dan objek – objek basis data lain berada disatu lingkungan (*environment*). Tiap lingkungan berisi satu katalog atau lebih, dari masing – masing katalog berisi satu himpunan skema (*schema*).

Skema adalah kumpulan objek basis data bernama yang saling berhubungan satu dengan lain (satu basis data dapat dideskripsikan di satu skema atau skema yang lain). Objek – objek dari skema bisa *berupa table, view, domain, assertion, collation, translation*, dan *character set*. Semua objek di satu skema mempunyai pemilik yang sama dan berbagi sejumlah *default.*

Berikut ini adalah *sintaks* untuk membuat *database* (baik di *MS SQL Server* ,*MySQL Server* ataupun *Oracle*:

```
Create Database [Database_name]
```

Sebagai contoh, misalkan kita akan membuat database baru dengan menggunakan *MS SQL Server* atau *MYSQL* dengan nama *database* NilaiMahasiswa pada bab 2, maka perintahnya adalah:

```
Create Database NilaiMahasiswa
```

Untuk melakukan penghapusan database pada *MS SQL Server* atau *MySQL*, sintaknya adalah:

```
Drop Database [Database_name]
```

Misalnya kita akan melakukan penghapusan pada *database* NilaiMahasiswa, maka perintahnya adalah:

```
Drop Database NilaiMahasiswa
```

Sedangkan untuk melakukan perubahan nama *database* pada *MS SQL Server*, sintaknya adalah:

```
Sp_RenameDB "[Database_lama], "[Database_baru]"
```

Misalnya kita akan melakukan perubahan nama *database* NilaiMahasiswa menjadi NilaiMHS, maka perintahnya adalah:

```
Sp_RenameDB "NilaiMahasiswa", "NilaiMHS"
```

b. Pembuatan *table*

Setelah dilakukan proses penciptaan *database*, kemudian kita dapat menciptakan struktur table untuk relasi – relasi pada basis data terebut.

Sintaks SQL untuk melakukan pembuatan tabel baru didalam basis data :

```
Create Table table_name
{ column_name data_type [NULL | NOT NULL]}
```

dimana :

table_name adalah nama tabel yang akan dibuat.

Column_name adalah nama-nama atribut yang akan terdapat di dalam tabel_name.

Data_type adalah domain nilai masing-masing atribut tersebut yang di tentukan berdasarkan tipe datanya.

NULL menspesifikasikan apakah kolom tersebut boleh kosong datanya.

NOT NULL menspesifikasikan kolom tersebut tidak boleh kosong (harus isi datanya) dan biasanya untuk identifikasi *primary key* pada table bentukan.

Pada perintah Create Table, minimal kita harus mendefinisikan nama tabel, kolom-kolom dan tipe datanya. Misalnya kita akan membuat tabel Mahasiswa pada database NilaiMahasiswa yang telah kita buat sebelumnya, dengan struktur table sebagai berikut:

Column Name	Data Type	Length	Allow Nulls
nim	char	9	
nama_m	varchar	35	
tpt_lhr_m	varchar	26	✓
tgl_lhr_m	datetime	8	✓
j_kelamin	varchar	10	✓
alm_m	varchar	90	✓
kota_m	varchar	20	✓
agama_m	varchar	10	✓
kode_jur	char	2	✓

Catatan : struktur table SQL Server

Tabel 10.3. Struktur table Mahasiswa

Untuk membuat perintah SQl untuk membuat table Mahasiswa tersebut adalah sebagai berikut:

1. Membuat table belum ada primary keynya.

```
Create Table  Mahasiswa
(
nim char (9)  not null,
nama_m  varchar (35) not null,
tpt_lhr_m varchar(26),
tgl_lhr_m  datetime,
j_kelamin varchar(10),
alm_m varchar(90),
kota_m  varchar (20),
agama_m varchar(10),
telpon_m char (13),
kode_jur char (2)
)
```

Perintah diatas adalah membuat *table* baru dengan nama Mahasiswa, dimana pada saat membuat *table* kita belum menentukan *primary key* pada table tersebut, untuk itu *field* nim harus

didefinisikan *not null.* Untuk membuat *primary key* setelah *table* terbentuk, akan tetapi belum dibuat *primary key* nya maka perintahnya adalah sebagai berikut :

```
Alter Table Mahasiswa
Add Constraint PkMahasiswa Primary Key(nim)
```

Perintah diatas adalah membuat *primary key* pada table Mahasiswa, di8mana *primary key* nya adalah nim, dengan nama *constraint*nya adalah PkMahasiswa. Untuk nama *constraint* ini tidak harus PkMahasiswa, kita bisa mendefinisikan dengan PkMHS atau lainnya.

2. Membuat table langsung dibentuk primary keynya.

Kita juga bisa membentuk table Mahasiswa tersebut dengan langsung membentuk *primary key*nya pada saat *mengcreate table* tersebut, perintahnya adalah:

```
Create Table  Mahasiswa
(
nim char (9)  Primary Key,
nama_m  varchar (35) not null,
tpt_lhr_m varchar(26),
tgl_lhr_m  datetime,
j_kelamin varchar(10),
alm_m varchar(90),
kota_m  varchar (20),
agama_m varchar(10),
telpon_m char (13),
kode_jur char (2)
)
```

atau :

```
Create Table  Mahasiswa
(
nim char (9),
nama_m  varchar (35) not null,
tpt_lhr_m varchar(26),
tgl_lhr_m  datetime,
j_kelamin varchar(10),
alm_m varchar(90),
kota_m  varchar (20),
agama_m varchar(10),
telpon_m char (13),
kode_jur char (2),
Constraint PkMHS Primary Key (nim)
)
```

c. Merubah struktur *table*

Dengan perintah ALTER TABLE kita dapat melakukan menambah kolom (ADD) pada table, menghapus kolom dan indeks (DROP).

1. Menambah kolom

   Misalkan kita akan menambahkan kolom pada table mahasiswa dengan nama kolom email, varchar (30) null, maka perintahnya adalah:

```
ALTER TABLE Mahasiswa
Add email varchar(30)
```

2. Merubah kolom

   Misalkan kita akan merubah kolom email tipe datanya diganti menjadi char(40) pada table mahasiswa, maka perintahnya adalah:

```
ALTER TABLE Mahasiswa
ALTER Column email Char(40)
```

3. Menghapus kolom

   Misalkan kita akan menghapus kolom email yang kita tambahkan pada table mahasiswa, maka perintahnya adalah:

```
ALTER TABLE Mahasiswa
DROP Column email
```

d. Penghapusan *table*

Setelah dilakukan proses penciptaan *table*, kita dapat melakukan penghapusan table baik dari sisi isi dan struktur tablenya. Terdapat DBMS yang melarang drop terhadap table yang masih berisi data, dengan demikian kita harus melakukan proses 2 (dua) langkah, yaitu:

- Kita harus mengosongkon isi *table* dengan mengggunakan perintah *DELETE*.
- Kita menghapus definisi *table* menggunakan perintah *DROP TABLE.*

Sintaks untuk melakukan penghapusan tble adalah sebagai berikut:

```
Drop Table [table_name] [RESTRICT | CASCADE]
```

Klausa Restrict atau Cascade mengacu pada apa yang terjadi pada objek – objek basis data lain yang bergantung pada table ini, misalkan VIEW. Restrict berarti akan tidak mengijinkan atau membatalkan drop table, sementara Cascade, maka objek – objek basis data yang bergantung pada table akan ikut terhapus juga. Restrict dan Cascade ini dapat saja tidak diimplementasikan oleh Vendor DBMS.

Misalkan kita akan melakukan penghapusan data pada table Mahasiswa maka perintahnya adalah:

```
Delete * From Mahasiswa
```

Atau :

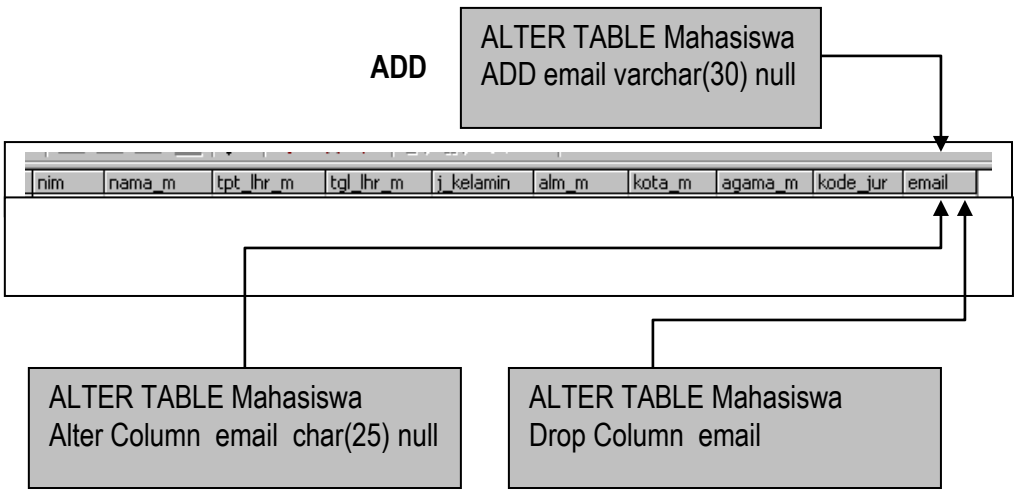
```
Delete from Mahasiswa
```

Untuk melakukan penghapusan table Mahasiswa berserta strukturnya, maka perintahnya adalah:

```
Drop Table Mahasiswa
```

e. Manipulasi terhadap definisi kolom

Penambahan (*insert*), penghapusan (*drop*), dan pengubahan (*update*) terhadap kolom (*field*) pada table dengan menggunakan perintah *ALTER TABLE.*



Gambar 10.1. Skema manipulasi definisi table

Keterangan :

ADD, pada perintah tersebut adalah menambahkan *field* baru dengan nama email, tipe data varchar(30) null pada *table* Mahasiswa.

UPDATE, pada perintah tersebut adalah merubah tipe data dan lebar untuk *field* email dengan tipe data dirubah menjadi char(25) null pada *table* Mahasiswa.

DROP, pada perintah tersebut adalah menghapus *field* email pada *table* Mahasiswa.

Kebanyakan vendor mengimplementasikan ALTER TABLE sebelum kalimat ini masuk dalam standard SQL. Kalimat ALTER TABLE baru dimasukkan kedalam SQL pada SQL92. terdapat beberapa perbedaan mengenai cara kerja kalimat ini antara vendor – vendor. Pada SQL92 ALTER TABLE dapat melakukan hal – hal berikut ini:

1. Menambahkan kolom kedalam *table.*
2. Menghapus kolom dari *table.*
3. Menambahkan *constraint* kedalam *table.*
4. Menghapus *constraint* dari *table.*
5. menambahkan nilai *default* kedalam kolom pada *table.*
6. menghapus nilai *default* dari kolom pada *table.*

ALTER TABLE sangat berharga saat kita perlu mendifinisi ulang *table*, namun kita seharusnya merancang *database* dengan sebesar mungkin tidak bergantung pada kalimat ini. Perubahan struktur *table* yang telah atau sedang digunakan operasional penuh berisiko bencana. *View* yang diturunkan dari table tidak berfungsi dan aplikasi berjalan tidak benar atau tidak berjalan sama sekali. Dengan demikian, kita seharusnya meranang *database* sehingga *ALTER TABLE* merupakan pilihan terakhir yang dapat digunakan.

Jika sistem tidak mendukung *ALTER TABLE*, atau jika kita akan menghindari penggunaannya, kita tinggal menciptakan *table* baru yang dikehendaki perubahannya dan mentransfer *table* lama kedalam *table* baru. Kemudian, kita memberikan wewenang pengaksesan pemakai *table* lama kedalam *table* baru.

10.3.4. Pembuatan Indeks

Indeks adalah struktur yang memberikan pengaksesan cepat ke baris – baris (*record*) pada *table* berdasarkan nilai satu kolom atau lebih. Keberadaan *indeks* dapat meningkatkan efisiensi

pengambilan data secara berarti. Tetapi karena *indeks* harus diperbarui sistem setiap kali terjadi perubahan *table* maka *indeks* juga memberi *overhead* (operasional) yang tinggi.

Pembentukan indeks tidak distandarkan, kebanyakan dialek mempunyai format berikut:

```
CREATE [UNIQUE] INDEX nama_indeks
ON nama_table (nama_kolom [ASC|DESC] [,…])
```

Nama_kolom, merupakan kunci indeks dimulai dari yang paling utama. Indeks hanya dapat iciptakan pada table – table dasar bukan view. Jika *option UNIQUE* digunakanm keunikan kolom atau kombinasi kolom yang diindeks akan dipakai sistem. Kemampuan ini diperlukan untuk kunci utama atau alternate key. Merupakan praktek bagus untuk membuat indeks – indeks unik setidaknya untuk kolom - kolom kunci utama pada saat table dasar dibuat dan sistem tidak secar otomatis memaksakan keunikan kunci utama.

Berikut ini contoh pembuatan indeks pada *table* Mahasiswa dimana kolom *indeksnya* adalah nim dengan nama *indeks* MhsIdx, maka perintahnya adalah:

```
CREATE INDEX MhsIdx
ON Mahasiswa
```

Bentuk *Index* pada atribute nim pada *table* Mahasiswa pada databse NilaiMahasiswa, sekali nim ditambahkan data baru, maka akan secara otomatis terurut datanya secara assending, setiap kali dilakukan *update, delete* atau *insert* data, oleh karena itu *Index* diperlukan .

Jika kita membuat indeks untuk table dasar dan kemudian ingin menghilangkan, dapat menggunakan pernyataan *DROP INDEX* dengan format sintaks sebagai berikut:

```
DROP INDEX  nama_index
```

Misalkan kita akan menghapus *indeks* yang telah dibuat sebelumnya dengan nama *indeks* MhsIdx, maka perintahnya adalah:

```
DROP INDEX  MhsIdx
```

10.3.5. Aturan Penamaan

MS SQL Server menyediakan aturan penamaan *identifier objek* dan penggunaan pembatas untuk identifier tidak *standard.* Kita disarankan menamai objek menggunakan identifier standard.

a. *Identifier standard.*

Identifier standard panjangnya 1 sampai 128 karakter dan dapat berisi huruf, simbol (@, #, _, or $), dan angka, spasi tidak diijinkan. Aturan penggunaan identifier adalah:

- Karakter pertama harus berupa karakter a-z atau A-Z atau salah satu simbol @, #, or _.
- Setelah karakter pertama, identtifier dapat huruf, angka, atau simbol @, #, _, aatau $.
- Nama identifier yang dimulai simbol mempunyai penggunaan khusus : identifier dimulai dengan @ menunjukkan variable atau parameter lokal, banyak identifier fungsi SQL Server dimulai @@ sehingga kita seharusnya tidak menciptakan identifier yang dimulai @@, identifier yang dimulai # menunjukkan table atau prosedur lokal, identifier yag dimulai ## menunjukkan objek temporer.

b. *Delimited Identifier.*

Jika identifier memenuhi semua aturan utuk format identifier, maka identifier dapat digunakan dengan atau tanpa delimiter. Jika identifier tidak memenuhi salah satu aturan atau lebih, maka identifier harus selalu diberi delimiter.

Delimited identifier dapat digunakan pada situasi – situasi berikut ini:

- Ketika nama berisi spasi atau karakter lain yang tidak diijinkan pada identifier standard.
- Ketika kata kunci digunakan untuk nama objek atau bagian – bagian dari nama objek.
- Delimated identifier diapit tanda kurung atau tanda petik ganda ketika digunakan pada kalimat Transact-SQL, contohnya:
  - Bracketed identifier diberi delimiter dengan tanda kurung siku ([ ])
  -

```
SELECT * FROM [Blanks In Table Name]
```

  - Quoted identifier diberi delimiter by double quotation marks (“ ”)

```
SELECT * FROM “Blanks In Table Name”
```

Ketika melakukan penamaan objek – objek basis data, kita seharusnya:

1. Agar menggunakan nama yang pendek.
2. Gunakan nama – nama yang berarti apabila memungkinkan.

3. Gunakan konvensi penamaan yang jelas dan sederhana. Putuskan apa yang terbaik untuk situasi yang dihadapi dan berlakulah konsisten, cobalah tidak membuat konvensi penamaan yang komplek karena akan sulit dimengerti.
4. Gunakan identifier yang membedakan tipe objek, khususnya untuk view dan store procedure. Administrator sering salah antara view dengan table, awali nama view name dengan vw.
5. Jaga agar nama – nama objek dan pemakai adalah unik .

10.4. SQL - DML

DML adalah subset SQL untuk melakukan manipulasi tupel – tupel pada basis data relaasional. DML mendefinisikan kalimat pengambilan, penyisipan, pembaruan dan penghapusan data. Pengembang akan sering menggunakan kalimat – kalimat DML dibandingkan kalimat – kalimat jenis ain.

10.4.1. Konsep DML

DML menyediakan 4 (empat) pernyataan untuk melakukan manipulasi data dalam database, yaitu:

1. *SELECT*, untuk *query (*meminta informasi) dari *database*.
2. *INSERT*, untuk melakukan penyisipan data pada table dalam suatu *database.*
3. *UPDATE*, untuk melakukan perubahan data pada suatu table dalam suatu *database.*
4. *DELETE*, untuk melakukan penghapusan data pada suatu table dalam suatu *database.*

SQL merupakan bahasa manipulasi data yang lengkap dan dapat digunakan untuk mengelola data basisdata. Perintah untuk melakukan modifikasi basis data tidak sekompleks kalimat SELECT yang digunakan untuk query. Pada sesi ini, kita mendeskripsikan tiga pernyataan SQL yaitu:

1. *INSERT*, untuk menambah baris baru kedalam *table.*
2. *UPDATE*, untuk memodifikasi (*update*) data yag telah ada pada *table.*
3. *DELETE*, untuk menghilangkan baris data pada suatu *table.*

Kalimat *SELECT* merupakan kalimat kompleks bila dibandingkan kalimat – kalimat DML yang lain. Pemahaman terhadap kalimat *SELECT* akan mempermudah pemahaman kalimat lain. Perintah *SELECT* mengambil data dari table tanpa mempengaruhi / mengubah data yang disimpan. Perintah ini hanya digunakan untuk mengirim atau mengambil data tanpa mengubahnya. Sementara itu, kalimat

UPDATE, INSERT dan *DELETE* digunakan untuk mempengaruhi / mengubah data pada table. Penggunaan yang tidak tepat dari perintah *UPDATE, INSERT* dan *DELETE* dapat menyebabkan kehilangan data atau terjadi korupsi data. Kalimat *UPDATE, INSERT* dan *DELETE* juga mempengaruhi / mengubah indeks. Sistem basis data akan mengubah indeks saat terjadi perubahan record atau penambahan data baru pada table.

Berikut ini adalah skema relasi untuk kasus SQL, yaitu pada database NilaiMahasiswa, beserta isi data dari masing – masing relasi:

Mahasiswa : (**nim**, nama_m, tpt_lahir_m, tgl_lhr_m, j_kelamin, alm_m, kota_m, agama_m, telpon_m, kode_jur)
Jurusan : **kode_jur**, nama_jur, jenjang, nama_kajur)
Matakuliah : (**kdmk**, nama_mk, sks, semester)
Dosen : (**nid**, nama_d, pendidikan_d, agama_d, alamat_d, kota_d, telpon_d
Nilai : (**nim, kdmk, smt**, nid, nilai_absen, nilai_tugas, nilai_uts, nilai_uas)

nim	nama_m	tpt_lhr_m	tgl_lhr_m	j_kelamin	alm_m	kota_m	agama_m	telpon_m	kode_jur
I01031001	Dewi Nurbaini	Bekasi	12/10/87	Wanita	Jl. Dahlia I Blok BC 2/3	Bekasi Utara	Hindu	021-8791290	TI
I01031002	Deni Hermawan	Jakarta	1/17/80	Pria	Jl. Seruni Raya No.3	Cikarang	Islam	021-9897119	TI
I01031003	Rini Hapsari	Bogor	8/18/81	Wanita	Jl. Cikarang Baru No.2	Cikarang	Islam	021-8911122	TI
I01031004	Sylvia Dwita Ningrum	Semarang	10/19/81	Wanita	Jl. Sultan Agung No.23	Bekasi Barat	Budha	021-8781112	TI
I01031005	Susilo Wahono	Jakarta	12/21/79	Pria	Jl. Kaliabang Tengah No.12	Jakarta Timur	Islam	021-8912782	TI
I01031006	Yonita Veronika	Bekasi	10/24/80	Wanita	Jl. Bunda Harapan No.36	Bekasi Barat	Kristen	021-8901112	TI
K01031001	Ahmad Sofyan	Surabaya	2/13/77	Pria	Jl. SMP I Nurul Huda No.4	Bekasi Timur	Islam	021-8789991	KA
K01031002	Daffa Alrozak	Solo	10/10/78	Pria	Jl. Muara Indah II No.1	Bekasi Barat	Islam	021-8781199	KA
M01031001	Lestary Ramiati	Bogor	1/10/76	Wanita	Jl. Telaga Murni I No.1	Bogor	Islam	0251-456767	MI
M01031002	Ani Lusiamah	Bandung	10/12/77	Wanita	Jl. Bulak Damai 3 No. 34	Cibinong	Islam	0251-343311	MI
S01031001	Ira Sulistyaningsih	Bekasi	12/1/79	Wanita	Jl. Purna 11 No. 7	Tambun	Budha	021-8912788	SI
S01031002	Yudi Ridwan	Subang	1/19/80	Pria	Jl. Kebun Kelapa I No.3	Cibitung	Hindu	021-8123412	SI
T01031001	Chandra Khirana	Cirebon	2/10/79	Pria	Jl. Masjid Al-Hidayah No.7	Bekasi Barat	Hindu	021-8712343	TK
T01031002	Fitria Choirunissa	Bekasi	1/21/79	Wanita	Jl. Gandaria Selatan I No.8	Jakarta Timur	Islam	021-8123111	TK
T01031003	Muhammad Fahrurozi	Jakarta	10/13/80	Pria	Jl. Dewi Sartika No.83	Jakarta Timur	Islam	021-8991222	TK
T01031004	Yuni Nurwati	Jakarta	11/14/78	Wanita	Jl. Pisangan Raya No.14	Bekasi Utara	Budha	021-8921221	TK

Tabel 10.4. Data value Mahasiswa

kode_jur	nama_jur	jenjang	nama_kajur
KA	Komputerisasi Akuntansi	Diploma 3	Rini Wulandari,MM,MMSi
MI	Manajemen Informatika	Diploma 3	Wahono Diprodjo, MM, MKom.
SI	Sistem Informasi	Strata 1	Bagus Hermansyah, MM.Si,M.Kom.
TI	Teknik Informatika	Strata 1	Fadjar Sasongko, MT, M.Kom.
TK	Teknik Komputer	Diploma 3	Agus Budiyantara, T, MT.

Column Name	Data Type	Length	Allow Nulls
kode_jur	char	2	
nama_jur	varchar	30	
jenjang	varchar	10	✓
nama_kajur	varchar	50	✓

Tabel 10.5. Data value dan struktur table Jurusan

kdmk	nama_mk	sks	semester
MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	1
MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	1
MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	1
MKK231203	MATEMATIKA DISKRIT I	2	1
MKK231205	AKUTANSI DASAR I	2	1
MKK231213	PENGANTAR EKONOMI	2	1
MPK131201	PENDIDIKAN AGAMA I	2	1
MPK131203	PENDIDIKAN PANCASILA	2	1
MPK131205	BAHASA INGGRIS I	2	1
MPK231207	SISTEM BASIS DATA	2	1
MPK131206	BAHASA INGGRIS II	2	2
MPK131204	PENDIDIKAN KEWARGANEGARAAN	2	2
MKK231206	AKUTANSI DASAR II	2	2
MKK231204	MATEMATIKA DISKRIT II	2	2
MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	2
MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	2
MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	2
MKB331205	PERANCANGAN BASIS DATA	4	2
MKB331206	PAKET PROGRAM APLIKASI II	1	2

Column Name	Data Type	Length	Allow Nulls
kdmk	char	9	
nama_mk	char	50	✓
sks	int	4	✓
semester	int	4	✓

Tabel 10.6. Data value dan struktur table Matakuliah

nid	nama_d	pendidikan_d	agama_d	alamat_d	kota_d	telpon_d
D01	Amir Mahmud,SE	S1	Islam	Jl. Nangka Permai No.17	Bekasi	021-8712311
D02	Susilo Wibowo, M.Kom	S2	Islam	Jl. Bambu Apus No.24	Bekasi	021-8723122
D03	Tukino Saputro,MM.Si	S2	Islam	Jl. Indah Kelana II No.10	Cikarang	021-8789991
D04	Rudiansyah,M.Kom.	S2	Kristen	Jl. A. Yani No.23	Bogor	0251-490111
D05	Sri Mulyani, Ir	S1	Hindu	Jl. Imam Bonjol No.2	Jakarta	021-3123411
D06	Didik Sanjaya, M.Kom.	S2	Islam	Jl. Anggrek No.392	Bekasi	021-8253212
D07	Primadina A,S.Kom.	S1	Islam	Jl. Kelana Jaya I No.5	Bekasi	021-8971221
D08	Djuwadi, MSc, M.Kom.	S2	Islam	Jl. Pejanten Raya No.1	Jakarta Selatan	021-7612322
D09	Suganda, Ir,MM.Si	S2	Islam	Jl. Pondok Senyum II No.2	Bekasi Timur	021-8981231

Column Name	Data Type	Length	Allow Nulls
nid	char	3	
nama_d	varchar	35	✓
pendidikan_d	varchar	10	✓
agama_d	varchar	10	✓
alamat_d	varchar	90	✓
kota_d	varchar	20	✓
telpon_d	char	13	✓

Tabel 10.7. Data value dan struktur table Dosen

nim	kdmk	smt	nid	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
M01031001	MPK131201	1	D02	12	78	89	55
M01031002	MPK131201	1	D02	14	67	78	87
K01031002	MPK131201	1	D02	12	87	56	56
T01031001	MPK131201	1	D02	14	78	98	89
I01031001	MPK131201	1	D02	14	89	78	89
I01031003	MPK131201	1	D02	13	87	56	67
S01031001	MPK131201	1	D02	14	67	54	78
S01031001	MPK131203	1	D05	14	76	67	88
I01031003	MPK131203	1	D05	13	77	45	78
I01031001	MPK131203	1	D05	14	90	67	78
T01031001	MPK131203	1	D05	12	78	88	78
K01031002	MPK131203	1	D05	13	56	78	78
M01031002	MPK131203	1	D05	11	89	90	90
M01031001	MPK131203	1	D05	13	77	87	76
M01031001	MPK131204	2	D01	11	56	78	67
M01031002	MPK131204	2	D01	10	78	90	76
I01031003	MPK131204	2	D01	11	67	78	89
S01031001	MPK131204	2	D01	9	78	57	80
K01031002	MPK131204	2	D01	13	67	66	89

Column Name	Data Type	Length	Allow Nulls
nim	char	9	
kdmk	char	9	
smt	int	4	
nid	char	3	✓
nilai_absen	int	4	✓
nilai_tugas	int	4	✓
nilai_uts	int	4	✓
nilai_uas	int	4	✓

Tabel 10.8. Data value dan struktur table Nilai

10.4.2. Kalimat Insert

Terdapat 2 (dua) bentuk pernyataan kalimat *INSERT*, yaitu:

1. Bentuk pertama *INSERT*, dimana memungkinkan satu baris tunggal disisipkan kealam table.
2. Bentuk kedua *INSERT*, dimana memungkinkan banyak baris sekaligus dikopikan kesatu table atau lebih.

Bentuk pertama *INSERT:*

Setelah Struktur tabel terbentuk, data dapat dimasukkan ke dalam tabel dengan perintah ***Insert*** dengan menggunakan *Query,* baik menggunakan database MS Access, SQL Server, MySQL ataupun Oracle.

Sintaks untuk melakukan insert data kealam suatu table adalah sebagai berikut:

INSERT [INTO] ***NAMA TABEL*** **[Daftar_Kolom] Value** ***DAFTAR_NILAI***

INSERT : Klausa ini menspesifikasikan nama tabel dimana data bisa ditambahkan

VALUE : Klausa ini menspesifikasikan nilai data yang akan disisipkan ke dalam kolom pada tabel

Daftar_Kolom : Merupakan daftar kolom yang dipisahkan oleh tanda koma menyatakan kolom-kolom yang akan diisi data. Jika tidak ada kolom yang dinyatakan, berarti semua kolom di dalam akan diisi data. Jika hanya sebagian daftar yang dinyatakan, nilai null atau nilai default akan diisikan ke kolom yang tidak disebutkan dalam daftar kolom.

Daftar Nilai : Daftar nilai untuk kolom tabel yang akan disisipkan sebagai sebuah baris data dalam tabel. Data yang diberikan pada daftar nilai harus sesuai dengan daftar kolom. Banyak data harus sama dengan banyak kolom, tipe data, presisi, dan skala dari setiap data harus sesuai dengan kolomnya.

Contoh : Misalkan kita akan menyisipkan data pada ke 5 (lima) table pada database NilaiMahasiswa tersebut diatas untuk masing – masing table 1 (satu) record untuk record pertama, maka perintahnya adalah:

1. Table Mahasiswa

```
INSERT INTO Mahasiswa
Values ('I01031001','Dewi Nurbaini','Bekasi','12/10/87',
'Wanita','Jl. Dahlia I Blok BC 2/3','Bekasi Utara',
'Hindu','021-8791290','TI')
```

2. Table Jurusan

```
INSERT INTO Jurusan
Values ('KA','Komputerisasi Akuntansi','Diploma 3',
'Rini Wulandari,MM,MMSi')
```

3. Table Matakuliah

```
INSERT INTO Matakuliah
Values ('MKB331201','PENGANTAR TEKNOLOGI INFORMASI',3,1)
```

4. Table Dosen

```
INSERT INTO Dosen
Values ('D01','Amir Mahmud,SE','S1','Islam',
'Jl. Nagka Permai No.17', 'Bekasi','021-8712311')
```

5. Table Nilai

```
INSERT INTO Nilai
Values ('M01031001','MPK131201,1,'D02',12,78,89,55)
```

Catatan :

1. Perhatikan untuk pemakaian tanda petik (' '), untuk tipe data *int,number,money*, dan lainnya yang bisa digunakan untuk proses kalkulasi tidak perlu ditambahkan tanda petik terebut.
2. Kita dapat memasukan seluruh record sekaligus jika menggunakan database MS-SQL Server, MySQL dan Oracle kecuali untuk MS-Access harus dimasukkan per record datanya.

Kita juga dapat melakukan pengisian data untuk kolom tertentu pada suatu *table*, sebagai contoh berikut ini adalah memasukkan data pada record kedua untuk masing – masing *table* diatas untuk kolom tertentu yang diisi datanya:

1. Table Mahasiswa

```
INSERT INTO Mahasiswa
(nim,nama_m,tpt_lhr_m,tgl_lhr_m,j_kelamin)
Values ('I01031002','Deni Hermawan','Jakarta','1/17/80',
'Pria')
```

Artinya adalah kita memasukkan data untuk kolom nim,nama_m,tpt_lhr_m,tgl_lhr_m,j_kelamin pada table Mahasiswa

2. Table Jurusan

```
INSERT INTO Jurusan (kode_jur,nama_jur,jenjang)
Values ('MI','Manajemen Informatika','Diploma 3')
```

3. Table Matkuliah

```
INSERT INTO Matakuliah (kdmk,nama_mk)
Values ('MKB331203','PRAKTIKUM PAKET PROGRAM APLIKASI I')
```

4. Table Dosen

```
INSERT INTO Dosen(nid,nama_d,pendidikan_d,agama_d,telpon_d)
Values ('D02','Susilo Wibowo,M.Kom','S2','Islam','021-
8723122')
```

5. Table Nilai

```
INSERT INTO Nilai (nim,kdmk,smt,nid,nilai_absen)
Values ('M01031002','MPK131201,1,'D02',14)
```

Catatan :

1. Untuk kolom *primary key table* dan kolom lainnya yang nilai datanya *Not Null* (harus isi), maka kita harus memasukkan datanya kalau tidak maka data yang kit insert tidak akan bisa.
2. Urutan nilai data (isi data) haarus sesuai dengan urutan kolom yang ada pada *table* tersebut.

Bentuk kedua *INSERT:*

Sintaks untuk melakukan *insert* untuk melakukan *copy data* kedalam suatu *table* adalah sebagai berikut:

```
INSERT INTO nama_table [(daftar_kolom)]
SELECT …
```

Nama_table dan daftar_kolom didefiniskan sebelum satu baris tunggal disisipkan. Klausa *SELECT* berupa sembarang pernyataan *SELECT* sah. Baris disisipkan kedalam table yang identik dan table hasil yang diproduksi subselect. Batasan pada bentuk pertama dapat diterapkan pada bentuk kedua ini.

Kalimat *INSERT* mempunyai batasan – batasan dalam penggunaannya, yaitu:

1. Kita harus selalu menspesifikasikan nilai – nilai untuk kunci utama dan kolom - kolom yang dinyatakan sebagai *NOT NULL.*
2. Data baru harus sesuai dengan tipe data untuk kolom yang dimaksud.
3. Ketika menggunakan ***foreign key***, kita harus memasukkan nilai – nilai absah pada table relasional.

RDBMS melakukan pemeriksaan untuk menjamin ketentuan – ketentuan diataas dipenuhi. Jika kita memberikan perintah kalimat insert yang tidak absah, RDBMS akan memberikan kesalahan dan tidak mengijinkan perubahan.

Sebagai contoh kita akan mengkopi (duplikasi) *table* baru dengan nama MahasiswaNew hasil copy dari table Mahasiswa pada database NilaiMahasiswa:

1. Kita harus membuat *table* baru dengan nama *table* MahasiswaNew

```
Create Table  MahasiswaNew
(
nimb char (9)  Primary Key,
nama_mb varchar (35) not null,
tpt_lhr_mb varchar(26),
tgl_lhr_mb  datetime,
j_kelaminb varchar(10),
alm_mb varchar(90)
)
```

2. Lalu lakukan pembuatan query berikut ini:

```
INSERT INTO MahasiswaNew  (nimb,nama_mb,tpt_lhr_mb,
tgl_lhr_mb, j_kelaminb,alm_mb)
SELECT nim,nama_m,tpt_lhr_m, tgl_lhr_m, j_kelamin,
alm_m FROM Mahasiswa
```

Artinya adalah, kita melakukan pemasukan data baru kedalam *table* MahasiswaNew, dimana dataya diambilkan dari data yang ada pada *table* Mahasiswa untuk kolom tertentu yang dilakukan *copy* datanya.

Contoh berikut ini adalah melakukan pengcopian (duplikasi data), dimana data yang dicopy adalah seluruh *record* dan seluruh kolom yang ada pada *table* tersebut.

```
SELECT *
INTO MatakuliahBaru
FROM Matakuliah
```

Artinya adalah, kita melakukan copy data beserta struktur tablenya untuk *table* Matakuliah kedalam tale MatakuliahBaru pada *database* NilaiMahasiswa.

Kita juga bisa melakukan *copy* pada suatu table ke dalam *table* yang baru, dimana yang dicopy adalah struktur *table* beserta isi datanya untuk 2 (dua) atau lebih *table* dalam suatu *database.*

Misalkan kita akan mengcopy *table* lama Mahasiswa dan Jurusan pada *database* NilaiMahasiswa kedalam *table* baru dengan nama MahasiswaJurusan, maka perintahnya adalah:

```
SELECT nim,nama_m,j_kelamin,alm_m,nama_jur,jenjang
INTO MahasiswaJurusan
FROM Mahasiswa inner join Jurusan on
Mahasiswa.kode_jur = Jurusan.kode_jur
```

Atau kita akan membuat duplikasi dengan nama MHSJurMI, dimana yang dicopy jurusannya adalah 'Manajemen Informatika, maka perintahnya adalah:

```
SELECT nim,nama_m,j_kelamin,alm_m,nama_jur,jenjang
INTO MahasiswaJurusan
FROM Mahasiswa inner join Jurusan on
Mahasiswa.kode_jur = Jurusan.kode_jur
Where nama_jur='Manajemen Informatika'
```

10.4.3. Kalimat Update

Kalimat *UPDATE* memungkinkan kita memodifikasi satu nilai kolom atau lebih unuk data *table* yang telah ada. Kita dapat menerapkan perubahan kesemua baris di *table*, satu subset baris, atau satu baris. Sintaks dari kalimat *UPDATE* tersebut adalah sebagai berikut:

```
UPDATE table_name
SET column_1 = value_1
[,column_2 = value_2 [,…]]
WHERE condition
```

Sebagaimana kalimat *INSERT*, beberapa batasan juga diterapkan terhadap kalimat *UPDATE* dan juga menjadi sasaran bataan – batasan pada *INSERT.* Kita dapat menggunakan kalusa *WHERE* untuk membatasi baris – baris yang dilakukan perubahan. Ini memungkinkan kita menerapkan perubahan – perubahan secara selektif. Update memungkinkan kita memodifikasi kolom – kolom yang berisi kunci utama atau foreign key sepanjang hasil perubahan memenuhi constraint integritas yang diberlakukan.

Pada sub bab 10.4.2 kita telah mencoba untuk memasukkan data dengan kalimat insert, pada insert record kedua ada beberapa kolom yang belum dimasukkan datanya, untk memasukkan data pada record kedua terebut adalah sebagai berikut:

1. Table Mahasiswa

```
UPDATE Mahasiswa set alm_m='Jl.Seruni Raya No.3',
kota_m='Cikarang',agama_m='Islam',telpon_m='02-9897119',
kode_jur='TI'
WHERE nim='I01031002'
```

2. Table Jurusan

```
UPDATE Jurusan set nama_kajur='Wahono Diprodjo,MM,MKom.'
WHERE kode_jur='MI'
```

3. Table Matakuliah

```
UPDATE Matakuliah set sks=1, semester=1
WHERE kdmk='MKB331203'
```

4. Table Dosen

```
UPDATE Dosen set alamat_d='Jl. Bambu Apus No.24',
kota_d='Bekasi
WHERE nid='D02'
```

5. Table Nilai

```
UPDATE Nilai set nilai_tugas=67, nilai_uts=78, nilai_uas=87
WHERE nim='M01031002',kdmk='MPK131201',smt=1
```

10.4.4. Kalimat Delete

Kalimat *DELETE* memungkinkan kita menghapus satu record ataulebih pada suatu table dlam database. Sintaks dari kalimat *DELETE* tersebut adalah sebagai berikut:

```
DELETE FROM table_name
WHERE condition
```

Kalimat *DELETE* hanya bisa dijalankan bila perubahan – perubahan yang ditimbulkan tidak membuat basis data melanggar integritas yang telah ditetapkan sebelumnya.

Table_name, dapat berupa *table* dasar atau *view* yang dapat diperbaharui, syarat adalah optional. Jika tidak ada klausa *WHERE*, maka semua baris akan terhapus dari *table.* Penghapusan semua baris di *table*

bukan berarti menghapus *table*, jika akan menghapus table dan definisi table maka menggunakan perintah *DROP TABLE*.

Jika klausa WHERE dispesifikasikan, maka hanya baris – baris yang memenuhi syarat akan dihapus.

1. Menghapus seluruh baris (record.).

   Misalkan kita akan menghapus seluruh record pada table MahasiswaNew yang telah kita buat dengan cara duplikasi (copy) sebelumnya, maka perintahnya adalah:

```
DELETE
FROM MahasiswaNew
```

2. Menghapus baris (record) tertentu.

   Misalkan kita akan menghapus record tertentu dimana sks nya adalah = 1, pada table MatakuliahBaru yang telah kita buat dengan cara duplikasi (copy) sebelumnya, maka perintahnya adalah:

```
DELETE
FROM MatakuliahBaru
Where sks=1
```

10.4.5. Kalimat Select

SQL menyediakan perintah *select* untuk mengakses dan mengeluarkan data dari *database server*. Dengan *sintaks* sebagai berikut :

Select [All | Distinct] *Pilih_daftar_kolom*
[Into [Nama_tabel baru]]
From *Nama_tabel / Nama View*
[Where Klausa]
[Group By Klausa]
[Having Klausa]
[Order By Klausa]
[Compute Klausa]

Keterangan :

Pilih_daftar_kolom : Menyatakan pilihan terhadap kolom atau atribut dari data yang dipilih.

Nama_tabel : Tabel yang akan diambil datanya.

a. Perintah Select untuk Memilih Semua Kolom

Perintah *select* untuk menampilkan semua data dan kolom pada suatu *table* pada *database*. sebagai contoh misalkan akan menampilkan data pada *table* Jurusan pada *database* NilaiMahasiswa, dapat menggunakan 2 (dua) cara:

1. Dengan menggunakan tanda (*)

```
SELECT *
From Jurusan
```

Hasil dari pernyataan query tersebut adalah:

	kode_jur	nama_jur	jenjang	nama_kajur
1	KA	Komputerisasi Akuntansi	Diploma 3	Rini Wulandari,MM,MMSi
2	MI	Manajemen Informatika	Diploma 3	Wahono Diprodjo, MM, MKom.
3	SI	Sistem Informasi	Strata 1	Bagus Hermansyah, MM.Si,M.Kom.
4	TI	Teknik Informatika	Strata 1	Fadjar Sasongko, MT, M.Kom.
5	TK	Teknik Komputer	Diploma 3	Agus Budiyantara, T, MT.

Tabel 10.9. Select * table Jurusan

2. Dengan memilih seluruh kolom

```
SELECT kode_jur,nama_jur,jenjang,nama_kajur
From Jurusan
```

Maka hasilnya sama seperti tampak pada table 10.9 diatas (jumlah kolom 4 (empat) dan record / barisnya adalah 5 (lima)

b. Menampilkan data dengan kolom/atribute tertentu

Untuk menampilkan data pada kolom tertentu pada suatu table dalam database sintaksnya adalah sebagai berikut:

```
Select  nama_kolom,[nama_kolom,….]
From  Nama_tabel
```

Berikut ini adalah akan menampilkan data dalam kolom nim, nama_m, alm_m, kota_m, telpon_m dari *table* Mahasiswa, maka perintah SQL nya adalah:

```
SELECT nim, nama_m, alm_m, kota_m, telpon_m
From Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah (5 (lima) kolom dan *record* / barisnya adalah 16 (enambelas), karena datanya hanya 16 (enambelas *record* pada *table* mahasiswa tersebut:

	nim	nama_m	alm_m	kota_m	telpon_m
1	MO1031001	Lestary Ramiati	Jl. Telaga Murni I No.1	Bogor	0251-456767
2	KO1031001	Ahmad Sofyan	Jl. SMP I Nurul Huda No.4	Bekasi Timur	021-8789991
3	MO1031002	Ani Lusiamah	Jl. Bulak Damai 3 No. 34	Cibinong	0251-343311
4	KO1031002	Daffa Alrozak	Jl. Muara Indah II No.1	Bekasi Barat	021-8781199
5	TO1031001	Chandra Khirana	Jl. Masjid Al-Hidayah No.7	Bekasi Barat	021-8712343
6	IO1031001	Dewi Nurbaini	Jl. Dahlia I Blok BC 2/3	Bekasi Utara	021-8791290
7	IO1031002	Deni Hermawan	Jl. Seruni Raya No.3	Cikarang	021-9897119
8	TO1031002	Fitria Choirunissa	Jl. Gandaria Selatan I No.8	Jakarta Timur	021-8123111
9	IO1031003	Rini Hapsari	Jl. Cikarang Baru No.2	Cikarang	021-8911122
10	IO1031004	Sylvia Dwita Ningrum	Jl. Sultan Agung No.23	Bekasi Barat	021-8781112
11	SO1031001	Ira Sulistyaningsih	Jl. Purna 11 No. 7	Tambun	021-8912788
12	TO1031003	Muhammad Fahrurozi	Jl. Dewi Sartika No.83	Jakarta Timur	021-8991222
13	IO1031005	Susilo Wahono	Jl. Kaliabang Tengah No.12	Jakarta Timur	021-8912782
14	IO1031006	Yonita Veronika	Jl. Bunda Harapan No.36	Bekasi Barat	021-8901112
15	SO1031002	Yudi Ridwan	Jl. Kebun Kelapa I No.3	Cibitung	021-8123412
16	TO1031004	Yuni Nurwati	Jl. Pisangan Raya No.14	Bekasi Utara	021-8921221

Gambar 10.10. *Query* untuk menampilkan data dengan kolom tertentu

c. Mengganti Nama Kolom

Ketika *Query* menampilkan hasil perintah *select*, nama kolom yang ditampilkan sesuai dengan nama kolom yang dispesifikasikan dalam *table*. Untuk mengganti nama kolom yang dihasilkan sesuai dengan keinginan kita, sintaks perintahnya adalah :

```
Select  nama_kolom  kolom alias ,[nama_kolom kolom alias]
From nama_tabel
```

Keterangan :

Kolom alias merupakan judul kolom yang diinginkan user.

Sebagai contoh, misalkan kita akan menampilkan nim, nama_m tampilan kolomnya menjadi ‘Nama Mahasiswa’, tpt_lhr_m menjadi ‘Tempat Lahir’, tgl_lhr_m menjadi ‘Tanggal Lahir dan telpon_m menjadi ‘Telpon’, maka perintah *query* nya adalah:

```
SELECT  nim,nama_m 'Nama Mahasiswa',tpt_lhr_m 'Tempat
        Lahir',tgl_lhr_m 'Tanggal Lahir',
         telpon_m 'Telpon'
FROM    Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah (5 (lima) kolom dan *record* / barisnya adalah 16 (enambelas), seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Tanggal Lahir	Telpon
1	M01031001	Lestary Ramiati	Bogor	1976-01-10...	0251-456767
2	K01031001	Ahmad Sofyan	Surabaya	1977-02-13...	021-8789991
3	M01031002	Ani Lusiamah	Bandung	1977-10-12...	0251-343311
4	K01031002	Daffa Alrozak	Solo	1978-10-10...	021-8781199
5	T01031001	Chandra Khirana	Cirebon	1979-02-10...	021-8712343
6	I01031001	Dewi Nurbaini	Bekasi	1987-12-10...	021-8791290
7	I01031002	Deni Hermawan	Jakarta	1980-01-17...	021-9897119
8	T01031002	Fitria Choirunissa	Bekasi	1979-01-21...	021-8123111
9	I01031003	Rini Hapsari	Bogor	1981-08-18...	021-8911122
10	I01031004	Sylvia Dwita Ningrum	Semarang	1981-10-19...	021-8781112
11	S01031001	Ira Sulistyaningsih	Bekasi	1979-12-01...	021-8912788
12	T01031003	Muhammad Fahrurozi	Jakarta	1980-10-13...	021-8991222
13	I01031005	Susilo Wahono	Jakarta	1979-12-21...	021-8912782
14	I01031006	Yonita Veronika	Bekasi	1980-10-24...	021-8901112
15	S01031002	Yudi Ridwan	Subang	1980-01-19...	021-8123412
16	T01031004	Yuni Nurwati	Jakarta	1978-11-14...	021-8921221

Gambar 10.11. *Query* untuk menampilkan data dengan kolom tertentu dan alias

d. Operator Aritmatika

Pernyatan SQL mendukung operator yang menampilkan operasi aritmatika seperti, penjumlahan, pengurangan, pembagian, perkalian, dan modulus pada kolom dengan jenis data numeric. Jenis data numeric adalah *INT, Smallint, Decimal, Numeric, Float, Real, Money, SmallMoney* dan lainnya tergantung DBMS yang digunakan.

Operator yang didukung oleh pernyataan SQL adalah :

- **+** untuk penjumlahan
- **-** untuk pengurangan
- **/** untuk pembagian
- ***** untuk perkalian
- **%** untuk modulus

Keterangan : semua operator aritmatika dapat digunakan dalam perintah Select .

d.1. Operator penjumlahan (+)

Tampilkan kdmk,nama_mk,sks dari table Matakuliah, dimana kolom kdmk diganti menjadi 'Kode Matakuliah, kolom nama_mk diganti menjadi 'Matakuliah' serta kolom sks + 2, maka perintahnya adalah:

```
SELECT  kdmk  'Kode  Matakuliah',  nama_mk  'Matakuliah',
        sks,sks+2 'SKS + 2'
FROM    Matakuliah
```

Hasil dari pernyataan *query* tersebut adalah (4 (lima) kolom dan *record* / barisnya adalah 19 (sembilan belas) karena data *table* matakuliah baru diisi 19 (sembilan belas) *record*, seperti tampak pada gambar berikut:

	Kode Matakuliah	Matakuliah	sks	SKS + 2
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	5
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	3
3	MKB331205	PERANCANGAN BASIS DATA	4	6
4	MKB331206	PAKET PROGRAM APLIKASI II	1	3
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	5
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	3
7	MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	4
8	MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	4
9	MKK231203	MATEMATIKA DISKRIT I	2	4
10	MKK231204	MATEMATIKA DISKRIT II	2	4
11	MKK231205	AKUTANSI DASAR I	2	4
12	MKK231206	AKUTANSI DASAR II	2	4

Gambar 10.12. Query dengan operator aritmatika penjumlahan

d.2. Operator pengurangan (-)

Tampilkan kdmk,nama_mk,sks dari table Matakuliah, dimana kolom kdmk diganti menjadi ’Kode Matakuliah, kolom nama_mk diganti menjadi ’Matakuliah’ serta kolom sks - 2, maka perintahnya adalah:

```
SELECT  kdmk  'Kode  Matakuliah',  nama_mk  'Matakuliah',
        sks,sks-1 'SKS-1'
FROM    Matakuliah
```

Hasil dari pernyataan *query* tersebut adalah (4 (lima) kolom dan *record* / barisnya adalah 19 (sembilan belas) karena data *table* matakuliah baru diisi 19 (sembilan belas) *record*, seperti tampak pada gambar berikut:

	Kode Matakuliah	Matakuliah	sks	SKS - 1
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	2
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	0
3	MKB331205	PERANCANGAN BASIS DATA	4	3
4	MKB331206	PAKET PROGRAM APLIKASI II	1	0
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	2
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	0
7	MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	1
8	MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	1
9	MKK231203	MATEMATIKA DISKRIT I	2	1
10	MKK231204	MATEMATIKA DISKRIT II	2	1
11	MKK231205	AKUTANSI DASAR I	2	1
12	MKK231206	AKUTANSI DASAR II	2	1
13	MKK231213	PENGANTAR EKONOMI	2	1

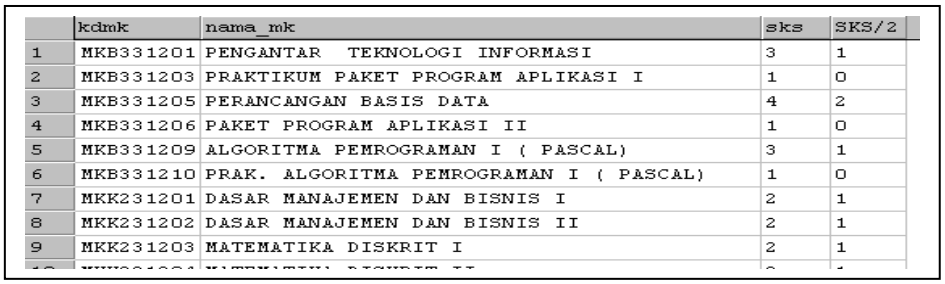
Gambar 10.13. Query dengan operator aritmatika pengurangan

d.3. Operator pembagian (/)

Tampilkan kdmk,nama_mk,sks dari table Matakuliah, kolom sks / 2, maka perintahnya adalah:

```
SELECT kdmk, nama_mk,sks,sks/2 'SKS/2'
FROM   Matakuliah
```

Hasil dari pernyataan *query* tersebut adalah (4 (lima) kolom dan *record* / barisnya adalah 19 (sembilan belas) karena data *table* matakuliah baru diisi 19 (sembilan belas) *record*, seperti tampak pada gambar berikut:



	kdmk	nama_mk	sks	SKS/2
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	1
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	0
3	MKB331205	PERANCANGAN BASIS DATA	4	2
4	MKB331206	PAKET PROGRAM APLIKASI II	1	0
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	1
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	0
7	MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	1
8	MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	1
9	MKK231203	MATEMATIKA DISKRIT I	2	1

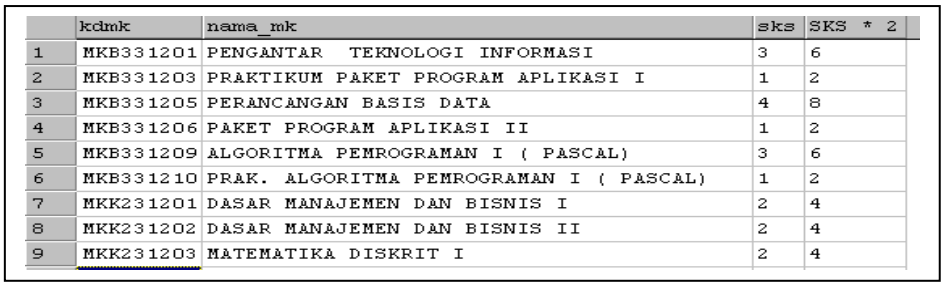
Gambar 10.14. Query dengan operator aritmatika pembagian

d.4. Operator perkalian (*)

Tampilkan kdmk,nama_mk,sks dari table Matakuliah, kolom sks * 2, maka perintahnya adalah:

```
SELECT kdmk, nama_mk,sks,sks * 2 'SKS * 2'
FROM   Matakuliah
```

Hasil dari pernyataan *query* tersebut adalah (4 (lima) kolom dan *record* / barisnya adalah 19 (sembilan belas) , seperti tampak pada gambar berikut:



	kdmk	nama_mk	sks	SKS * 2
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	6
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	2
3	MKB331205	PERANCANGAN BASIS DATA	4	8
4	MKB331206	PAKET PROGRAM APLIKASI II	1	2
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	6
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	2
7	MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	4
8	MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	4
9	MKK231203	MATEMATIKA DISKRIT I	2	4

Gambar 10.14. Query dengan operator aritmatika perkalian

d.5. Operator modulus (%)

Tampilkan kdmk,nama_mk,sks dari table Matakuliah, kolom sks % 2, maka perintahnya adalah:

```
SELECT kdmk, nama_mk,sks,sks % 2 'SKS % 2'
FROM   Matakuliah
```

Hasil dari pernyataan *query* tersebut adalah (4 (lima) kolom dan *record* / barisnya adalah 19 (sembilan belas) , seperti tampak pada gambar berikut:

	kdmk	nama_mk	sks	SKS % 2
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3	1
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	1
3	MKB331205	PERANCANGAN BASIS DATA	4	0
4	MKB331206	PAKET PROGRAM APLIKASI II	1	1
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3	1
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1	1
7	MKK231201	DASAR MANAJEMEN DAN BISNIS I	2	0
8	MKK231202	DASAR MANAJEMEN DAN BISNIS II	2	0
9	MKK231203	MATEMATIKA DISKRIT I	2	0
10	MKK231204	MATEMATIKA DISKRIT II	2	0

Gambar 10.15. Query dengan operator aritmatika modulus (sisa bagi)

e. Menyeleksi Baris dengan Kondisi

Adakalanya hanya beberapa baris saja yang perlu diretrieved dari sebuah *table.* Clausa *Where* disediakan oleh SQL untuk menspesifikasikan kondisi tersebut. Sintaks dari pernyataan tersebut adalah:

```
Select   Daftar_kolom
From     nama_tabel
Where    kondisi
```

Pada metode, klausa *where* dapat dibagi dalam beberapa kategori seperti berikut:

1. Comparison operator : seperti =,>,<,>=,<=,!=/<>
2. Range operator : seperti Between dan Not Beetween
3. List operator : seperti In dan Not In
4. String operator : seperti Like dan Not Like
5. Logical Operator : seperti And, Or, Not.

e.1. Operator Comparison

Pada operator comparison operator ini dapat dikelompokkan menjadi operator = (sama dengan), > (lebih besar), < (lebih kecil), >= (lebih besar sama dengan), <= (lebih kecil sama dengan), != / <> (tidak sama dengan).

1. Operator =

1.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_uas>80, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_uas=80
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 1 (satu) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	S01031001	MPK131204	9	78	57	80

Gambar 10.16. Query Comparison operator = untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai),, dimana nilai_uts=78, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
       Mahasiswa.nim INNER    JOIN    Matakuliah    ON
       Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uts=78
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 5 (lima) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN KEWARGANEGARAAN	2	78	67
2	Ani Lusiamah	PENDIDIKAN AGAMA I	2	78	87
3	Daffa Alrozak	PENDIDIKAN PANCASILA	2	78	78
4	Dewi Nurbaini	PENDIDIKAN AGAMA I	2	78	89
5	Rini Hapsari	PENDIDIKAN KEWARGANEGARAAN	2	78	89

Gambar 10.17. Query Comparison operator = untuk join table

2. Operator >

2.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_uas>88, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_uas>88
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 5 (lima) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	I01031001	MPK131201	14	89	78	89
2	I01031003	MPK131204	11	67	78	89
3	K01031002	MPK131204	13	67	66	89
4	M01031002	MPK131203	11	89	90	90
5	T01031001	MPK131201	14	78	98	89

Gambar 10.18. Query Comparison operator > untuk 1 table

2.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai),, dimana nilai_uts>89, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
       Mahasiswa.nim INNER    JOIN    Matakuliah    ON
       Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uts>89
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 3 (tiga), seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uts	nilai_uas
1	Ani Lusiamah	PENDIDIKAN PANCASILA	2	90	90
2	Ani Lusiamah	PENDIDIKAN KEWARGANE...	2	90	76
3	Chandra Khirana	PENDIDIKAN AGAMA I	2	98	89

Gambar 10.19. Query Comparison operator > untuk join table

3. Operator <

3.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_uas<70, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_uas<70
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	IO1031003	MPK131201	13	87	56	67
2	KO1031002	MPK131201	12	87	56	56
3	MO1031001	MPK131201	12	78	89	55
4	MO1031001	MPK131204	11	56	78	67

Gambar 10.20. Query Comparison operator < untuk 1 table

3.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai),, dimana nilai_uts<80, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
       Mahasiswa.nim INNER    JOIN    Matakuliah    ON
       Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uts<65
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 5 (lima), seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uts	nilai_uas
1	Daffa Alrozak	PENDIDIKAN AGAMA I	2	56	56
2	Rini Hapsari	PENDIDIKAN AGAMA I	2	56	67
3	Rini Hapsari	PENDIDIKAN PANCASILA	2	45	78
4	Ira Sulistyaningsih	PENDIDIKAN AGAMA I	2	54	78
5	Ira Sulistyaningsih	PENDIDIKAN KEWARGANEGARAAN	2	57	80

Gambar 10.21. Query Comparison operator < untuk join table

4. Operator >=

4.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_uas>90, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_uas>=90
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	M01031002	MPK131203	11	89	90	90

Gambar 10.22. Query Comparison operator >= untuk 1 table

4.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai), dimana nilai_uts>=85, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
       Mahasiswa.nim INNER    JOIN    Matakuliah    ON
       Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uts>=85
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 6 (enam), seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN AGAMA I	2	89	55
2	Lestary Ramiati	PENDIDIKAN PANCASILA	2	87	76
3	Ani Lusiamah	PENDIDIKAN PANCASILA	2	90	90
4	Ani Lusiamah	PENDIDIKAN KEWARGANEGARAAN	2	90	76
5	Chandra Khirana	PENDIDIKAN AGAMA I	2	98	89
6	Chandra Khirana	PENDIDIKAN PANCASILA	2	88	78

Gambar 10.23. Query Comparison operator >= untuk join table

5. Operator <=

5.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_uas<=70, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_uas<=70
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	IO1031003	MPK131201	13	87	56	67
2	KO1031002	MPK131201	12	87	56	56
3	MO1031001	MPK131201	12	78	89	55
4	MO1031001	MPK131204	11	56	78	67

Gambar 10.24. Query Comparison operator <= untuk 1 table

5.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai), dimana nilai_uts>=85, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
       Mahasiswa.nim INNER    JOIN    Matakuliah    ON
       Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uts<65
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 5 (lima), seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uts	nilai_uas
1	Daffa Alrozak	PENDIDIKAN AGAMA I	2	56	56
2	Rini Hapsari	PENDIDIKAN AGAMA I	2	56	67
3	Rini Hapsari	PENDIDIKAN PANCASILA	2	45	78
4	Ira Sulistyaningsih	PENDIDIKAN AGAMA I	2	54	78
5	Ira Sulistyaningsih	PENDIDIKAN KEWARGANEGARAAN	2	57	80

Gambar 10.25. Query Comparison operator <= untuk join table

6. Operator != /<>

6.a. Untuk 1 (satu) *table*

Tampilkan nim,kdmk,nilai_absen,nilai_tugas,nilai_uts, nilai_uas dari table Nilai, dimana nilai_absen<>13 and nilai_absen !=14, maka perintah nya adalah:

```
SELECT    nim,kdmk,nilai_absen,nilai_tugas,nilai_uts,
          nilai_uas
FROM      Nilai
WHERE     nilai_absen<>13 and nilai_absen!=14
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom dan *record* / barisnya adalah 8 (delapan) , seperti tampak pada gambar berikut:

	nim	kdmk	nilai_absen	nilai_tugas	nilai_uts	nilai_uas
1	I01031003	MPK131204	11	67	78	89
2	K01031002	MPK131201	12	87	56	56
3	M01031001	MPK131201	12	78	89	55
4	M01031001	MPK131204	11	56	78	67
5	M01031002	MPK131203	11	89	90	90
6	M01031002	MPK131204	10	78	90	76
7	S01031001	MPK131204	9	78	57	80
8	T01031001	MPK131203	12	78	88	78

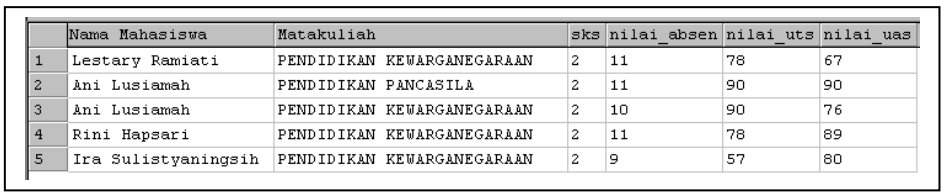
Gambar 10.24. Query Comparison operator <> untuk 1 table

6.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai), dimana nilai_uts>=85, maka perintah nya adalah:

```
SELECT  nama_m  'Nama  Mahasiswa',nama_mk  'Matakuliah',
        sks,nilai_absen,nilai_uts,nilai_uas
FROM  Nilai  INNER  JOIN  Mahasiswa  ON  Nilai.nim  =
      Mahasiswa.nim INNER    JOIN    Matakuliah    ON
      Nilai.kdmk=Matakuliah.kdmk
WHERE   nilai_absen<>13   and   nilai_absen!=14   and
      nilai_absen<>12
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 5 (lima), seperti tampak pada gambar berikut:



	Nama Mahasiswa	Matakuliah	sks	nilai_absen	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN KEWARGANEGARAAN	2	11	78	67
2	Ani Lusiamah	PENDIDIKAN PANCASILA	2	11	90	90
3	Ani Lusiamah	PENDIDIKAN KEWARGANEGARAAN	2	10	90	76
4	Rini Hapsari	PENDIDIKAN KEWARGANEGARAAN	2	11	78	89
5	Ira Sulistyaningsih	PENDIDIKAN KEWARGANEGARAAN	2	9	57	80

Gambar 10.25. Query Comparison operator <> untuk join table

e.2. Range Operator

Operator Range digunakan untuk meretrieved data yang dapat diperoleh dalam range tersebut, operator tersebut meliputi Between dan Not Between.

Sintaks untuk operator range tersebut adalah:

Select *Daftar_kolom*
From *nama_tabel*
Where *nama_kolom* [not] Between *expresi1 and expresi 2*

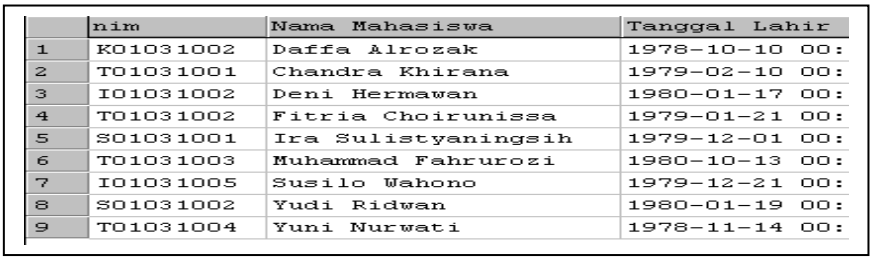
1. Operator Between

1.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tgl_lhr_m untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tgl_lhr_m diganti 'Tanggal Lahir' dimana mahasiswa yang tanggal lahirnya antara '11/17/1977' sampai '10/19/1980, maka perintah nya adalah:

```
SELECT  nim,nama_m  'Nama  Mahasiswa',tgl_lhr_m  'Tanggal
       Lahir'
FROM Mahasiswa
Where tgl_lhr_m BETWEEN '11/17/1977' and '10/19/1980'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 9 (sembilan) , seperti tampak pada gambar berikut:



	nim	Nama Mahasiswa	Tanggal Lahir
1	KO1031002	Daffa Alrozak	1978-10-10 00:
2	TO1031001	Chandra Khirana	1979-02-10 00:
3	IO1031002	Deni Hermawan	1980-01-17 00:
4	TO1031002	Fitria Choirunissa	1979-01-21 00:
5	SO1031001	Ira Sulistyaningsih	1979-12-01 00:
6	TO1031003	Muhammad Fahrurozi	1980-10-13 00:
7	IO1031005	Susilo Wahono	1979-12-21 00:
8	SO1031002	Yudi Ridwan	1980-01-19 00:
9	TO1031004	Yuni Nurwati	1978-11-14 00:

Gambar 10.26. Query Range operator Between untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk,sks (dari table matakuliah), nilai_uts,nilai_uas (dari table nilai), dimana nilai_uts nya antara 79 sampai 90, maka perintah nya adalah:

```
SELECT  nama_m,nama_mk,sks,nilai_uts,nilai_uas
FROM    Nilai INNER JOIN Mahasiswa ON Nilai.nim =
        Mahasiswa.nim INNER JOIN Matakuliah ON
        Nilai.kdmk=Matakuliah.kdmk
WHERE   nilai_uts BETWEEN 79 and 90
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 5 (lima) , seperti tampak pada gambar berikut:

	nama_m	nama_mk	sks	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN AGAMA I	2	89	55
2	Lestary Ramiati	PENDIDIKAN PANCASILA	2	87	76
3	Ani Lusiamah	PENDIDIKAN PANCASILA	2	90	90
4	Ani Lusiamah	PENDIDIKAN KEWARGANEGARAAN	2	90	76
5	Chandra Khirana	PENDIDIKAN PANCASILA	2	88	78

Gambar 10.27. Query Range operator Between untuk join table

2. Operator Not Between

2.a. Untuk 1 (satu) *table*

Tampilkan kdmk,nama_mk,sks dari table matakuliah, untuk kolom kdmk diganti menjadi 'Kode Matakuliah' kolom nama_mk diganti 'Matakuliah'. Dimana matakuliah yang sks nya bukan antara 1 dan 2, maka perintah nya adalah:

```
SELECT kdmk 'Kode Matakuliah',nama_mk 'Matakuliah',sks
FROM   Matakuliah
WHERE  sks Not BETWEEN 1 and 2
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 3 (tiga) , seperti tampak pada gambar berikut:

	Kode Matakuliah	Matakuliah	sks
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3
2	MKB331205	PERANCANGAN BASIS DATA	4
3	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3

Gambar 10.28. Query Range operator Not Between untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_mk (dari table matakuliah), nilai_uas (dari table nilai), dimana kolom nama_m diganti menjadi 'Nama Mahasiswa', kolom nama_mk diganti 'Matakuliah' dan nilai_uas nya bukan antara 70 sampai 90, maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',nama_mk 'Matakuliah',
     sks,nilai_uas
FROM Nilai INNER JOIN Mahasiswa ON Nilai.nim
     =Mahasiswa.nim
INNER JOIN Matakuliah ON Nilai.kdmk=Matakuliah.kdmk
WHERE nilai_uas NOT BETWEEN 70 and 90
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Matakuliah	sks	nilai_uas
1	Lestary Ramiati	PENDIDIKAN AGAMA I	2	55
2	Lestary Ramiati	PENDIDIKAN KEWARGANEGARAAN	2	67
3	Daffa Alrozak	PENDIDIKAN AGAMA I	2	56
4	Rini Hapsari	PENDIDIKAN AGAMA I	2	67

Gambar 10.29. Query Range operator Not Between untuk join table

e.3. List Operator

Operator List digunakan untuk menampilkan data yang dapat diperoleh dalam daftar (batasan) tertentu, operator tersebut meliputi In dan Not In.

Sintaks untuk operator list tersebut adalah:

Select ***Daftar_kolom***
From ***nama_tabel***
Where ***nama_kolom*** **operator List (‘*Daftar_value*’)**

1. Operator In

1.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tpt_lhr_m,telpon_m untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tpt_lhr_m diganti 'Tempat Lahir', kolom telpon_m diganti 'Telepon'. Dimana mahasiswa yang tempat lahirnya di Bogor, Surabaya dan Solo, maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa',tpt_lhr_m 'Tempat
       Lahir',telpon_m 'Telepon'
FROM   Mahasiswa
WHERE  tpt_lhr_m IN ('Bogor','Surabaya','Solo')
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Telepon
1	M01031001	Lestary Ramiati	Bogor	0251-456767
2	K01031001	Ahmad Sofyan	Surabaya	021-8789991
3	K01031002	Daffa Alrozak	Solo	021-8781199
4	I01031003	Rini Hapsari	Bogor	021-8911122

Gambar 10.30. Query List operator In untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_jur,jenjang (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom nama_jur diganti 'Jurusan', untuk mahasiswa yang nama_jur nya adalah 'Sistem Informasi' dan 'Teknik Informatika', maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',nama_jur 'Jurusan',
       jenjang
FROM   Mahasiswa   INNER   JOIN   Jurusan   ON   Mahasiswa.
       kode_jur=Jurusan.kode_jur
WHERE nama_jur IN ('Sistem Informasi',
       'Teknik Informatika')
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 8 (delapan) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Jurusan	jenjang
1	Dewi Nurbaini	Teknik Informatika	Strata 1
2	Deni Hermawan	Teknik Informatika	Strata 1
3	Rini Hapsari	Teknik Informatika	Strata 1
4	Sylvia Dwita Ningrum	Teknik Informatika	Strata 1
5	Ira Sulistyaningsih	Sistem Informasi	Strata 1
6	Susilo Wahono	Teknik Informatika	Strata 1
7	Yonita Veronika	Teknik Informatika	Strata 1
8	Yudi Ridwan	Sistem Informasi	Strata 1

Gambar 10.31. Query Range operator In untuk join table

2. Operator In

2.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tpt_lhr_m,telpon_m untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tpt_lhr_m diganti 'Tempat Lahir', kolom telpon_m diganti 'Telepon'. Dimana mahasiswa yang tempat lahirnya Bukan di Bekasi, Bogor dan Jakarta, maka perintah nya adalah:

```
SELECT  nim,nama_m  'Nama  Mahasiswa',tpt_lhr_m  'Tempat
     Lahir', telpon_m 'Telepon'
FROM   Mahasiswa
WHERE  tpt_lhr_m NOT IN ('Bekasi','Bogor','Jakarta')
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Telepon
1	KO1031001	Ahmad Sofyan	Surabaya	021-8789991
2	MO1031002	Ani Lusiamah	Bandung	0251-343311
3	KO1031002	Daffa Alrozak	Solo	021-8781199
4	TO1031001	Chandra Khirana	Cirebon	021-8712343
5	IO1031004	Sylvia Dwita Ningrum	Semarang	021-8781112
6	SO1031002	Yudi Ridwan	Subang	021-8123412

Gambar 10.32. Query List operator Not In untuk 1 table

2.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_jur (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom nama_jur diganti 'Jurusan', untuk mahasiswa yang nama_jur nya adalah bukan 'Sistem Informasi', 'Teknik Informatika' dan 'Manajemen Informatika', maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',nama_jur 'Jurusan'
FROM   Mahasiswa INNER JOIN Jurusan ON
       Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE  nama_jur NOT IN ('Sistem Informasi',
       'Teknik Informatika','Manajemen Informatika')
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Jurusan
1	Ahmad Sofyan	Komputerisasi Akuntansi
2	Daffa Alrozak	Komputerisasi Akuntansi
3	Chandra Khirana	Teknik Komputer
4	Fitria Choirunissa	Teknik Komputer
5	Muhammad Fahrurozi	Teknik Komputer
6	Yuni Nurwati	Teknik Komputer

Gambar 10.33. Query Range operator Not In untuk join table

e.4. String Operator

Key Word ***Like*** digunakan untuk memilih baris-baris yang sesuai dengan karakter yang digunakan. ***Like*** menggunakan karakter Wildcard yang bisa digunakan sebagai expresi.

% Wildcard	Deskripsi
%	String Karakter
_ Garis Bawah	Karakter Tunggal
[ ]	Karakter tunggal dalam range tertentu
[^]	Karakter tunggal yang tidak dalam range tertentu

1. Wildcard %

1.a. Untuk 1 (satu) *table*

a. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya untuk 2 huruf pertama adalah 'De', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE 'De%'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 2(dua) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	IO1031001	Dewi Nurbaini
2	IO1031002	Deni Hermawan

Gambar 10.34. Query String operator Wildcard % untuk 1 table (a)

b. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya untuk 2 huruf terakhirnya adalah 'An', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '%An'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 2(dua) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	K01031001	Ahmad Sofyan
2	I01031002	Deni Hermawan
3	S01031002	Yudi Ridwan

Gambar 10.35. Query String operator Wildcard % untuk 1 table (b)

c. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya mengandung huruf 'An', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '%An%'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	K01031001	Ahmad Sofyan
2	M01031002	Ani Lusiamah
3	T01031001	Chandra Khirana
4	I01031002	Deni Hermawan
5	S01031001	Ira Sulistyaningsih
6	S01031002	Yudi Ridwan

Gambar 10.36. Query String operator Wildcard % untuk 1 table (c)

2. Wildcard _

2.a. Untuk 1 (satu) *table*

a. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya setelah 3 huruf pertama mengandung huruf 'a', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '___a%'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 2(dua) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	KO1031001	Ahmad Sofyan
2	TO1031003	Muhammad Fahrurozi

Gambar 10.37. Query String operator Wildcard _ untuk 1 table (a)

b. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya sebelum 2 huruf terakhir mengandung huruf 'i', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '%i__'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 2(dua) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	IO1031001	Dewi Nurbaini
2	IO1031006	Yonita Veronika

Gambar 10.35. Query String operator Wildcard _ untuk 1 table (b)

3. Wildcard []

3.a. Untuk 1 (satu) *table*

a. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya pada huruf pertama mengandung huruf 'a' atau 'd', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '[ad]%'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 5(lima) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	KO1031001	Ahmad Sofyan
2	MO1031002	Ani Lusiamah
3	KO1031002	Daffa Alrozak
4	IO1031001	Dewi Nurbaini
5	IO1031002	Deni Hermawan

Gambar 10.37. Query String operator Wildcard [] untuk 1 table (a)

b. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya 1 huruf terakhir mengandung huruf 'i', 'k' atau 'a', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '%[ika]'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 9 (sembilan) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	MO1031001	Lestary Ramiati
2	KO1031002	Daffa Alrozak
3	TO1031001	Chandra Khirana
4	IO1031001	Dewi Nurbaini
5	TO1031002	Fitria Choirunissa
6	IO1031003	Rini Hapsari
7	TO1031003	Muhammad Fahrurozi
8	IO1031006	Yonita Veronika
9	TO1031004	Yuni Nurwati

Gambar 10.35. Query String operator Wildcard [] untuk 1 table (b)

4. Wildcard ^

4.a. Untuk 1 (satu) *table*

a. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya pada huruf pertama bukan mengandung huruf 'd', 'f', 'c', 'a' dan 'y', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '[^dfcay]%'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	M01031001	Lestary Ramiati
2	I01031003	Rini Hapsari
3	I01031004	Sylvia Dwita Ningrum
4	S01031001	Ira Sulistyaningsih
5	T01031003	Muhammad Fahrurozi
6	I01031005	Susilo Wahono

Gambar 10.37. Query String operator Wildcard ^ untuk 1 table (a)

b. Tampilkan nim,nama_m, untuk kolom nama_m diganti 'Nama Mahasiswa', dimana nama mahasiswanya 1 huruf terakhir tidak mengandung huruf 'i', 'n', 'a' atau 'k', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa'
FROM Mahasiswa
WHERE nama_m LIKE '%[^inak]'
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa
1	M01031002	Ani Lusiamah
2	I01031004	Sylvia Dwita Ningrum
3	S01031001	Ira Sulistyaningsih
4	I01031005	Susilo Wahono

Gambar 10.35. Query String operator Wildcard ^ untuk 1 table (b)

e.5. Logical Operator

Operator Logika yang umum disediakan didalam SQL adalah sebagai berikut:

OR Memberikan nilai True bila kondisi yang dispesifikasikan True

NOT Membalikkan nilai expresi yang mengikutinya

AND Memberikan hasil True bila semua kondisi yang dispesifikasikan True.

Sintaks untuk operator logika terebut adalah:

```
Select   daftar_kolom
From     nama_tabel
Where conditional expresi   {AND/OR} [Not]   conditional
expresi
```

1. Operator OR

Opertor ini digunakan untuk menguji apakah salah satu kedua ekspresi logika yang diberikan memiliki nilai TRUE, dengan aturan penulisan:

```
Ekspriesi1 OR Ekspresi2
```

Operator OR pada SQL akan menghasilkan nilai sesuai dengan table berikut:

Ekspresi1	Ekspresi2	Hasil
True	True	True
True	False	True
True	Null	True
False	True	True
False	False	False
False	Null	False
Null	True	True
Null	False	False
Null	Null	Null

1.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tpt_lhr_m,alm_m,kota_m, untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tpt_lhr_m diganti 'Tempat Lahir', kolom alm_m diganti 'Alamat' dan kolom kota_m diganti 'Kota'. Dimana mahasiswa yang tempat lahirnya di 'Bekasi' atau kota alamatnya adalah 'Cikarang', maka perintah nya adalah:

```
SELECT  nim,nama_m  'Nama  Mahasiswa',tpt_lhr_m  'Tempat
     Lahir', alm_m 'Alamat',kota_m 'Kota'
FROM   Mahasiswa
WHERE  tpt_lhr_m='Bekasi' OR kota_m='Cikarang'
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Alamat	Kota
1	IO1031001	Dewi Nurbaini	Bekasi	Jl. Dahlia I Blok BC 2/3	Bekasi Utara
2	IO1031002	Deni Hermawan	Jakarta	Jl. Seruni Raya No.3	Cikarang
3	TO1031002	Fitria Choirunissa	Bekasi	Jl. Gandaria Selatan I No.8	Jakarta Timur
4	IO1031003	Rini Hapsari	Bogor	Jl. Cikarang Baru No.2	Cikarang
5	SO1031001	Ira Sulistyaningsih	Bekasi	Jl. Purna 11 No. 7	Tambun
6	IO1031006	Yonita Veronika	Bekasi	Jl. Bunda Harapan No.36	Bekasi Barat

Gambar 10.36. Query Logical operator OR untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m,j_kelamin (dari table mahasiswa), nama_jur,jenjang (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom j_kelamin diganti 'Jenis Kelamin, kolom nama_jur diganti 'Jurusan', untuk mahasiswa yang j_kelamin nya 'Wanita' atau nama_jur = 'Teknik Informatika', maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',j_kelamin 'Jenis
Kelamin',nama_jur 'Jurusan',jenjang
FROM Mahasiswa INNER JOIN Jurusan ON
Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE j_kelamin='Wanita' OR nama_jur='Teknik
Informatika'
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 11 (sebelas) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Jenis Kelamin	Jurusan	jenjang
1	Lestary Ramiati	Wanita	Manajemen Informatika	Diploma 3
2	Ani Lusiamah	Wanita	Manajemen Informatika	Diploma 3
3	Dewi Nurbaini	Wanita	Teknik Informatika	Strata 1
4	Deni Hermawan	Pria	Teknik Informatika	Strata 1
5	Fitria Choirunissa	Wanita	Teknik Komputer	Diploma 3
6	Rini Hapsari	Wanita	Teknik Informatika	Strata 1
7	Sylvia Dwita Ningrum	Wanita	Teknik Informatika	Strata 1
8	Ira Sulistyaningsih	Wanita	Sistem Informasi	Strata 1
9	Susilo Wahono	Pria	Teknik Informatika	Strata 1
10	Yonita Veronika	Wanita	Teknik Informatika	Strata 1
11	Yuni Nurwati	Wanita	Teknik Komputer	Diploma 3

Gambar 10.37. Query Logical operator OR untuk join table

2. Operator NOT

Opertor ini digunakan untuk mendapatkan nilai kebalikan dari suatu logika atau ekspresi. Dengan demikian akan dihasilkan nilai TRUE hanya jika nilai ekspresi bernilai

FALSE, serta nilai akan menghasilkan NULL jika ekspresi memiliki nilai NULL, dengan aturan penulisan:

```
NOT Ekspresi
```

Operator NOT pada SQL akan menghasilkan nilai sesuai dengan table berikut:

Ekspresi	Hasil
True	False
False	True
Null	Null

2.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tpt_lhr_m,alm_m,kota_m, untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tpt_lhr_m diganti 'Tempat Lahir', kolom alm_m diganti 'Alamat' dan kolom kota_m diganti 'Kota'. Dimana mahasiswa yang tempat lahirnya bukan di 'Bekasi' ,'Cikarang,'Jakarta' dan 'Bogor', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa',tpt_lhr_m 'Tempat
     Lahir',alm_m 'Alamat',kota_m 'Kota'
FROM Mahasiswa
WHERE    tpt_lhr_m    NOT    in    ('Bekasi','Cikarang',
     'Jakarta','Bogor')
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Alamat	Kota
1	K01031001	Ahmad Sofyan	Surabaya	Jl. SMP I Nurul Huda No.4	Bekasi Timur
2	M01031002	Ani Lusiamah	Bandung	Jl. Bulak Damai 3 No. 34	Cibinong
3	K01031002	Daffa Alrozak	Solo	Jl. Muara Indah II No.1	Bekasi Barat
4	T01031001	Chandra Khirana	Cirebon	Jl. Masjid Al-Hidayah No.7	Bekasi Barat
5	I01031004	Sylvia Dwita Ningrum	Semarang	Jl. Sultan Agung No.23	Bekasi Barat
6	S01031002	Yudi Ridwan	Subang	Jl. Kebun Kelapa I No.3	Cibitung

Gambar 10.38. Query Logical operator NOT untuk 1 table

1.b. Untuk join *table*

Tampilkan nama_m (dari table mahasiswa), nama_jur,jenjang (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom nama_jur diganti 'Jurusan',

untuk mahasiswa yang nama_jurusannya bukan 'Teknik Informatika' dan 'Teknik Komputer', maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa', nama_jur
'Jurusan',jenjang
FROM Mahasiswa INNER JOIN Jurusan ON
Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE nama_jur NOT in('Teknik Informatika','Teknik
Komputer')
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Jurusan	jenjang
1	Lestary Ramiati	Manajemen Informatika	Diploma 3
2	Ahmad Sofyan	Komputerisasi Akuntansi	Diploma 3
3	Ani Lusiamah	Manajemen Informatika	Diploma 3
4	Daffa Alrozak	Komputerisasi Akuntansi	Diploma 3
5	Ira Sulistyaningsih	Sistem Informasi	Strata 1
6	Yudi Ridwan	Sistem Informasi	Strata 1

Gambar 10.39. Query Logical operator NOT untuk join table

3. Operator AND

Opertor ini digunakan untuk menguji beberapa ekspresi logika yang diberikan memiliki nilai TRUE, dengan aturan penulisan:

```
Ekspriesi1 AND Ekspresi2
```

Operator AND pada SQL akan menghasilkan nilai sesuai dengan table berikut:

Ekspresi1	Ekspresi2	Hasil
True	True	True
True	False	False
False	True	False
False	False	False
Null	True/False/Null	Null
True/False/Null	Null	Null

3.a. Untuk 1 (satu) *table*

Tampilkan nim,nama_m,tpt_lhr_m,j_kelamin untuk kolom nama_m diganti 'Nama Mahasiswa', kolom tpt_lhr_m diganti 'Tempat Lahir' dan kolom j_kelamin diganti 'Jenis Kelamin'. Dimana mahasiswa yang tempat lahirnya di 'Bekasi' dan jenis kelaminnya aalah 'Wanita', maka perintah nya adalah:

```
SELECT nim,nama_m 'Nama Mahasiswa',tpt_lhr_m 'Tempat
     Lahir',j_kelamin 'Jenis Kelamin'
FROM Mahasiswa
WHERE tpt_lhr_m='Bekasi' AND j_kelamin='Wanita'
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nim	Nama Mahasiswa	Tempat Lahir	Jenis Kelamin
1	I01031001	Dewi Nurbaini	Bekasi	Wanita
2	T01031002	Fitria Choirunissa	Bekasi	Wanita
3	S01031001	Ira Sulistyaningsih	Bekasi	Wanita
4	I01031006	Yonita Veronika	Bekasi	Wanita

Gambar 10.40. Query Logical operator AND untuk 1 table

3.b. Untuk join *table*

Tampilkan nama_m,j_kelamin (dari table mahasiswa), nama_jur,jenjang (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom j_kelamin diganti 'Jenis Kelamin', kolom nama_jur diganti 'Jurusan', untuk mahasiswa yang jenis kelaminnya 'Wanita' dan nama_jurusannya 'Teknik Informatika' dan 'Teknik Komputer', maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',j_kelamin 'Jenis
Kelamin',nama_jur 'Jurusan',jenjang
FROM Mahasiswa INNER JOIN Jurusan ON
Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE j_kelamin='Wanita' AND nama_jur IN('Teknik
Informatika','Teknik Komputer')
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 6 (enam) , seperti tampak pada gambar berikut:

	Nama Mahasiswa	Jenis Kelamin	Jurusan	jenjang
1	Dewi Nurbaini	Wanita	Teknik Informatika	Strata 1
2	Fitria Choirunissa	Wanita	Teknik Komputer	Diploma 3
3	Rini Hapsari	Wanita	Teknik Informatika	Strata 1
4	Sylvia Dwita Ningrum	Wanita	Teknik Informatika	Strata 1
5	Yonita Veronika	Wanita	Teknik Informatika	Strata 1
6	Yuni Nurwati	Wanita	Teknik Komputer	Diploma 3

Gambar 10.41. Query Logical operator AND untuk join table

f. Menguji Nilai Null

Unknown Values atau lebih dikenal dengan Null, akan diberikan apabila tidak ada input data. Nilai Null dapat ditampilkan dari tabel dengan menggunakan kunci **Is Null** dalam Klausa Where.

Sintaks untuk ekspresi ini adalah sebagai berikut:

```
Select  Daftar_kolom
From    nama_tabel
Where   nama_kolom . operator_Unknown_Values
```

f.1. Untuk 1 (satu) *table*

Tampilkan seluruh kolom pada table Mahasiswa, dimana untuk kolom telepon berisi data NULL., maka perintah nya adalah:

```
SELECT *
FROM Mahasiswa
WHERE telpon_m IS NULL
```

Hasil dari pernyataan *query* tersebut adalah 10 (sepuluh) kolom dan *record* / barisnya adalah 0 (nol) karena tidak ada kolom telpon_m berisi NULL, seperti tampak pada gambar berikut:

nim	nama_m	tpt_lhr_m	tgl_lhr_m	j_kelamin	alm_m	kota_m	agama_m	telpon_m	kode_jur

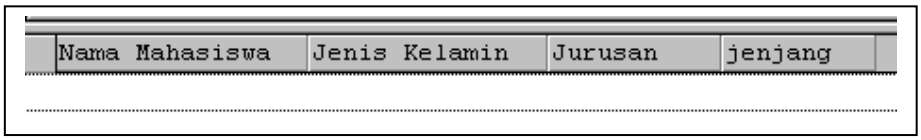
Gambar 10.42. Query Logical operator NULL untuk 1 table

f.2. Untuk join *table*

Tampilkan nama_m,j_kelamin (dari table mahasiswa), nama_jur,jenjang (dari table jurusan), dimana kolom nama_m diganti 'Nama Mahasiswa', kolom j_kelamin diganti 'Jenis Kelamin', kolom nama_jur diganti 'Jurusan'. Dimana pada kolom jenjang berisi NULL, maka perintah nya adalah:

```
SELECT nama_m 'Nama Mahasiswa',j_kelamin 'Jenis
Kelamin',nama_jur 'Jurusan',jenjang
FROM Mahasiswa INNER JOIN Jurusan ON
Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE jenjang IS NULL
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 0 (nol) karena tidak ada kolom jenjang yang nilainya NULL, seperti tampak pada gambar berikut:



Gambar 10.43. Query Logical operator NULL untuk join table

g. Fungsi Agregate

Terdapat beberapa fngsi agregate atau fungsi untuk melakukan suatu operasi terhadap himpunan data yang dapat digunakan dalam perintah query. Hasil dari penggunaan fungsi ini merupakan nilai perolehan yang dapat diolah lebih lanjut. Fungsi Agregate yang tersedia sebagai berikut :

Fungsi	Penjelasan
SUM	Menjumlahkan Nilai
MIN	Mencari nilai minimum
MAX	Mencari Nilai Maximum
AVG	Mencari Nilai Rata rata
COUNT	Menghitung jumlah data

Fungsi AVG, SUM, MIN, MAX dan COUNT meng *Ignore* nilai *Null* sedangkan fungsi COUNT() menghitung nilai Null.

1. Fungsi SUM

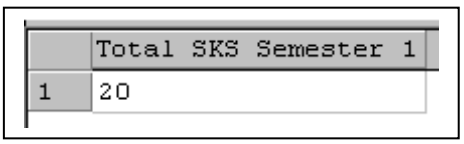
Fungsi ini mempunyai bentuk SUM(X), digunakan untuk menghasilkan nilai penjumlahan dari suatu data bilangan, dimana X adalah nilai numerik atau nama field yang memiliki nilai numerik atau rumus yang menghasilkan nilai numerik.

1.a. Untuk 1 (satu) *table*

Tampilkan kolom 'Total SKS Semester 1' dari table Matakuliah dan hitung jumlah sks nya pada semester 1, maka perintah nya adalah:

```
SELECT SUM(sks) 'Total SKS Semester 1'
From Matakuliah
Where semester=1
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu) , seperti tampak pada gambar berikut:



	Total SKS Semester 1
1	20

Gambar 10.44. Query Fungsi SUM untuk 1 table

2. Fungsi MIN

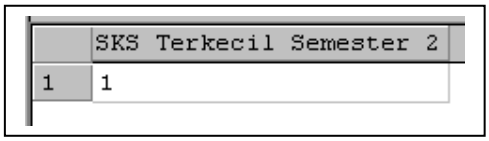
Fungsi ini mempunyai bentuk MIN(X), digunakan untuk menghasilkan nilai terkecil dari suatu data bilangan, dimana X adalah nilai numerik atau nama field yang memiliki nilai numerik atau rumus yang menghasilkan nilai numerik.

2.a. Untuk 1 (satu) *table*

Tampilkan kolom 'SKS Terkecil Semester 2' dari table Matakuliah pada semester 2, maka perintah nya adalah:

```
SELECT MIN(sks) 'SKS Terkecil Semester 2'
From Matakuliah
Where semester=2
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu), seperti tampak pada gambar berikut:



	SKS Terkecil Semester 2
1	1

Gambar 10.45. Query Fungsi MIN untuk 1 table

3. Fungsi MAX

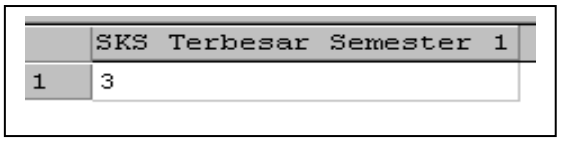
Fungsi ini mempunyai bentuk MAX(X), digunakan untuk menghasilkan nilai terbesar dari suatu data bilangan, dimana X adalah nilai numerik atau nama field yang memiliki nilai numerik atau rumus yang menghasilkan nilai numerik.

3.a. Untuk 1 (satu) *table*

Tampilkan kolom 'SKS Terbesar Semester 1' dari table Matakuliah pada semester 1, maka perintah nya adalah:

```
SELECT MAX(sks) 'SKS Terbesar Semester 1'
From Matakuliah
Where semester=1
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu), seperti tampak pada gambar berikut:



	SKS Terbesar Semester 1
1	3

Gambar 10.46. Query Fungsi MAX untuk 1 table

4. Fungsi AVG

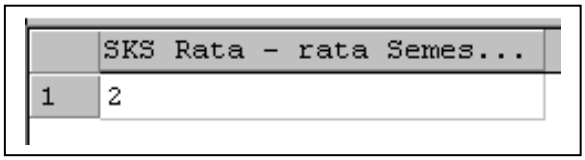
Fungsi ini mempunyai bentuk AVG(X), digunakan untuk menghasilkan nilai rata - rata dari suatu data bilangan, dimana X adalah nilai numerik atau nama field yang memiliki nilai numerik atau rumus yang menghasilkan nilai numerik. Nilai perhitungan AVG adalah nilai seluruh data dibagi dengan jumlah data, apabila terdapat nilai NULL pada satu atau lebih data, nilai NULL tersebut tidak digunakan dalam perhitungan dan jumlah data tidak ditambahkan.

4.a. Untuk 1 (satu) *table*

Tampilkan kolom 'SKS Rata – rata Semester 2' dari table Matakuliah pada semester 2, maka perintah nya adalah:

```
SELECT AVG(sks) 'SKS Rata - rata Semester 2'
From Matakuliah
Where semester=2
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu), seperti tampak pada gambar berikut:



	SKS Rata - rata Semes...
1	2

Gambar 10.47. Query Fungsi AVG untuk 1 table

5. Fungsi COUNT

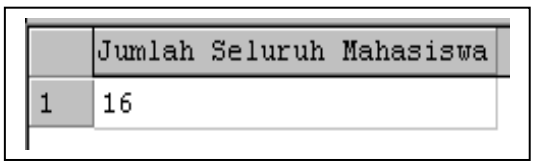
Fungsi ini mempunyai bentuk COUNT(field), digunakan untuk menghitung banyaknya data dalam suatu table dari hasil query. Dengan fungsi ini dapat dihitung jumlah data yang diperoleh atas dasar field tertentu atau seluruh field pada query. Dengan menyertakan nama field, nilai NULL pada field yang disebutkan tidak akan dimasukkan dalam kalkulasi.

5.a. Untuk 1 (satu) *table*

Tampilkan kolom 'Jumlah Seluruh Mahasiswa' dari table Mahasiswa, maka perintah nya adalah:

```
SELECT 'Jumlah Seluruh Mahasiswa'=count(nim)
From Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu), seperti tampak pada gambar berikut:



	Jumlah Seluruh Mahasiswa
1	16

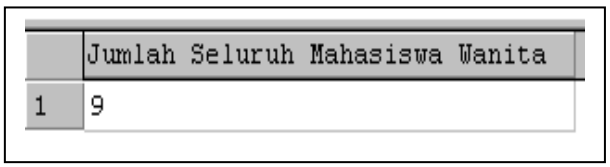
Gambar 10.48. Query Fungsi COUNT untuk 1 table (a)

5.b. Untuk 1 (satu) *table*

Tampilkan kolom 'Jumlah Seluruh Mahasiswa Wanita' dari table Mahasiswa dimana j_kelamin nya 'Wanita', maka perintah nya adalah:

```
SELECT 'Jumlah Seluruh Mahasiswa Wanita'=count(nim)
From Mahasiswa
Where j_kelamin='Wanita'
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 1 (satu), seperti tampak pada gambar berikut:



	Jumlah Seluruh Mahasiswa Wanita
1	9

Gambar 10.49. Query Fungsi COUNT untuk 1 table (b)

h. Fungsi Tanggal dan Waktu

Fungsi tanggal merupakan fungsi yang diperuntukkan bagi data, variable atau field yang berkaitan dengan data tanggal. Demikian pula untuk fungsi waktu yaitu fungsi yang diperuntukkan bagi data, variable atau field yang berkaitan dengan data waktu. Fungsi tanggal dan waktu dapat pula diterapkan untuk data dari tanggal sistem komputer yang sedang aktif.

Beberapa fungsi tanggal antara lain :

Fungsi	Sintaks	Keterangan
DATEADD	(Datepart, number, Date)	Menambah datepart ke tanggal sesuai dengan besar number
DATEDIFF	(Datepart, date, date2)	Mengkalkulasi angka datepart antara dua tanggal
DATEPART	(Datepart, date)	Mengembalikan datepart dari daftar tanggal sebagai integer
DATENAME	(Datepart, date)	Mengembalikan datepart dari daftar tanggal sebagai nilai Asci (contoh October)
GetDate	( )	Tanggal dan Waktu sekarang

Komponen tanggal yang disebut ***Datepart*** digunakan untuk menentukan elemen nilai tanggal untuk Aritmatika tanggal. Datepart dapat berupa :

Datepart		Singkatan	Nilai	Keteangan
Year	Tahun	YY	1753-9999	8246 Thn
Quarter	Kwartel	QQ	1-4	4 Kwartal 1thn
Month	Bulan	MM	1-12	12 bln setahun
Day Of Year	Hari ke	DY	1-366	366 hari setahun

Day	Tanggal	DD	1-31	31 hari sebulan
Week	Minggu	WW	0-51	52 mingg se thn
Weekday	Hari	DW	1-7 (1=Sunday)	7 hari seminggu
Hour	Jam	HH	0-23	24 jam sehari
Minute	Menit	MI	(0-59)	60 mnt sejam
Second	Detik	SS	(0-59)	60 detik semenit
Milisecond	Milidetik	MS	(0-999)	1000 mili sedetik

1. Fungsi DateAdd

Fungsi ini mempunyai bentuk DateAdd(DatePart,Number,Date), merupakan fungsi yang menambah datepart ke tanggal sesuai dengan besar number yang diberikan (DD,MM,YY).

1.a. DD

Tampilkan kolom nama_m,tgl_lhr_m, dan kolom ’30 Hari Setelah Tanggal Lahir’ dari table Mahasiswa dimana mahasiswa yang tempat lahirnya ’Bogor’, maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,"30 Hari Setelah Tanggal
Lahir"=DATEADD(dd,30,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m=’Bogor’
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 2 (dua) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	30 Hari Setelah Tanggal Lahir
1	Lestary Ramiati	1976-01-10 ...	1976-02-09 00:00:00.000
2	Rini Hapsari	1981-08-18 ...	1981-09-17 00:00:00.000

Gambar 10.50. Query Fungsi DadeAdd DD

1.b. MM

Tampilkan kolom nama_m,tgl_lhr_m, dan kolom ’5 Bulan Setelah Tanggal Lahir’ dari table Mahasiswa dimana mahasiswa yang tempat lahirnya ’Jakarta’, maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,"5 Bulan Setelah Tanggal
Lahir"=DATEADD(mm,5,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m='Jakarta'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	5 Bulan Setelah Tanggal Lahir
1	Deni Hermawan	1980-01-17 00:00:00.000	1980-06-17 00:00:00.000
2	Muhammad Fahrurozi	1980-10-13 00:00:00.000	1981-03-13 00:00:00.000
3	Susilo Wahono	1979-12-21 00:00:00.000	1980-05-21 00:00:00.000
4	Yuni Nurwati	1978-11-14 00:00:00.000	1979-04-14 00:00:00.000

Gambar 10.51. Query Fungsi DadeAdd MM

1.c. YY

Tampilkan kolom nama_m,tgl_lhr_m, dan kolom ’5 Tahun Setelah Tanggal Lahir’ dari table Mahasiswa dimana mahasiswa yang tempat lahirnya ’Jakarta’ dan ’Bekasi’, maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,"5 Tahun Setelah Tanggal
Lahir"=DATEADD(yy,5,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m in ('Jakarta',’Bekasi’)
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 8 (delapan) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	5 Tahun Setelah Tanggal Lahir
1	Dewi Nurbaini	1987-12-10 00:00:00.000	1992-12-10 00:00:00.000
2	Deni Hermawan	1980-01-17 00:00:00.000	1985-01-17 00:00:00.000
3	Fitria Choirunissa	1979-01-21 00:00:00.000	1984-01-21 00:00:00.000
4	Ira Sulistyaningsih	1979-12-01 00:00:00.000	1984-12-01 00:00:00.000
5	Muhammad Fahrurozi	1980-10-13 00:00:00.000	1985-10-13 00:00:00.000
6	Susilo Wahono	1979-12-21 00:00:00.000	1984-12-21 00:00:00.000
7	Yonita Veronika	1980-10-24 00:00:00.000	1985-10-24 00:00:00.000
8	Yuni Nurwati	1978-11-14 00:00:00.000	1983-11-14 00:00:00.000

Gambar 10.52. Query Fungsi DadeAdd YY

2. Fungsi DateIIF

Fungsi ini mempunyai bentuk DateDIFF(DatePart,Date,Date2), merupakan fungsi yang melakukan kalkulasi angka datepart antara dua tanggal yang diberikaan (DD,MM,YY).

2.a. DD

Tampilkan kolom nama_m,tgl_lhr_m, 'Tanggal dan Waktu Sekarang', 'Jumlah Hari' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Jakarta', maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,getdate() 'Tanggal dan
Waktu       Sekarang',"Jumlah       Hari"=DATEDIFF
(dd,tgl_lhr_m,getdate())
FROM Mahasiswa
Where tpt_lhr_m='Jakarta'
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	Tanggal dan Waktu Sekarang	Jumlah Hari
1	Deni Hermawan	1980-01-17 00:00:00.000	2006-06-07 12:59:16.043	9638
2	Muhammad Fahrurozi	1980-10-13 00:00:00.000	2006-06-07 12:59:16.043	9368
3	Susilo Wahono	1979-12-21 00:00:00.000	2006-06-07 12:59:16.043	9665
4	Yuni Nurwati	1978-11-14 00:00:00.000	2006-06-07 12:59:16.043	10067

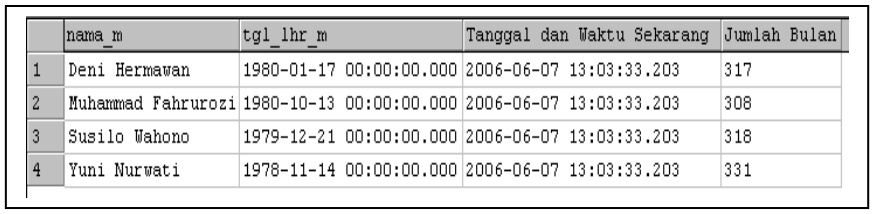
Gambar 10.53. Query Fungsi DadeDIIF DD

2.b. MM

Tampilkan kolom nama_m,tgl_lhr_m, 'Tanggal dan Waktu Sekarang', 'Jumlah Bulan' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Jakarta', maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,getdate() 'Tanggal dan
Waktu       Sekarang',"Jumlah       Bulan"=DATEDIFF
(mm,tgl_lhr_m,getdate())
FROM Mahasiswa
Where tpt_lhr_m='Jakarta'
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:



	nama_m	tgl_lhr_m	Tanggal dan Waktu Sekarang	Jumlah Bulan
1	Deni Hermawan	1980-01-17 00:00:00.000	2006-06-07 13:03:33.203	317
2	Muhammad Fahrurozi	1980-10-13 00:00:00.000	2006-06-07 13:03:33.203	308
3	Susilo Wahono	1979-12-21 00:00:00.000	2006-06-07 13:03:33.203	318
4	Yuni Nurwati	1978-11-14 00:00:00.000	2006-06-07 13:03:33.203	331

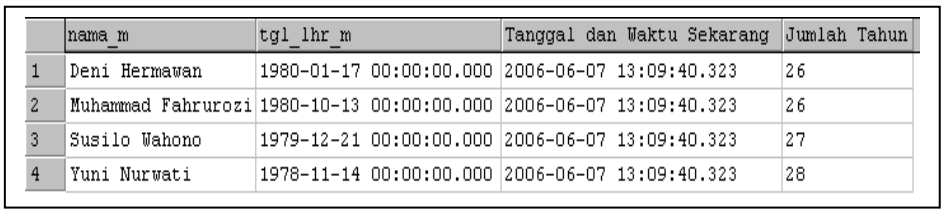
Gambar 10.54. Query Fungsi DadeDIFF MM

2.c. YY

Tampilkan kolom nama_m,tgl_lhr_m, 'Tanggal dan Waktu Sekarang', 'Jumlah Tahun' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Jakarta', maka perintah nya adalah:

```
SELECT nama_m, tgl_lhr_m,getdate() 'Tanggal dan
Waktu Sekarang',"Jumlah Tahun"=DATEDIFF
(yy,tgl_lhr_m,getdate())
FROM Mahasiswa
Where tpt_lhr_m='Jakarta'
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:



	nama_m	tgl_lhr_m	Tanggal dan Waktu Sekarang	Jumlah Tahun
1	Deni Hermawan	1980-01-17 00:00:00.000	2006-06-07 13:09:40.323	26
2	Muhammad Fahrurozi	1980-10-13 00:00:00.000	2006-06-07 13:09:40.323	26
3	Susilo Wahono	1979-12-21 00:00:00.000	2006-06-07 13:09:40.323	27
4	Yuni Nurwati	1978-11-14 00:00:00.000	2006-06-07 13:09:40.323	28

Gambar 10.55. Query Fungsi DadeDIFF YY

3. Fungsi DatePart

Fungsi ini mempunyai bentuk DatePart(DatePart,Date), merupakan fungsi yang mengembalikan datepart dari daftar tanggal sebagai integer sesuai tanggal yang diberikaan (DD,MM,YY).

3.a. DD

Tampilkan kolom nama_m,tgl_lhr_m, 'Tanggal Lahir' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Bogor', maka perintah nya adalah:

```
SELECT          nama_m,          tgl_lhr_m,"Tanggal
Lahir"=DATEPART(dd,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m='Bogor'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 2 (dua) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	Tanggal Lahir
1	Lestary Ramiati	1976-01-10 00:00:00.000	10
2	Rini Hapsari	1981-08-18 00:00:00.000	18

Gambar 10.56. Query Fungsi DatePart DD

3.b. MM

Tampilkan kolom nama_m,tgl_lhr_m, 'Bulan Lahir' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Bogor', maka perintah nya adalah:

```
SELECT          nama_m,          tgl_lhr_m,"Bulan
Lahir"=DATEPART(mm,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m='Bogor'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 2 (dua) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	Bulan Lahir
1	Lestary Ramiati	1976-01-10 00:00:00.000	1
2	Rini Hapsari	1981-08-18 00:00:00.000	8

Gambar 10.57. Query Fungsi DadePart MM

3.c. YY

Tampilkan kolom nama_m,tgl_lhr_m, 'Tahun Lahir' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Bogor', maka perintah nya adalah:

```
SELECT          nama_m,          tgl_lhr_m,"Tahun
Lahir"=DATEPART(yy,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m='Bogor'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 2 (dua) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	Tahun Lahir
1	Lestary Ramiati	1976-01-10 00:00:00.000	1976
2	Rini Hapsari	1981-08-18 00:00:00.000	1981

Gambar 10.58. Query Fungsi DadePart YY

4. Fungsi DateName

Fungsi ini mempunyai bentuk DateName(DatePart,Date), merupakan fungsi yang mengembalikan datepart dari daftar tanggal sebagai nilai ASCII (misal October), funsi ini hanya bergungsi untuk bulan (MM).

Tampilkan kolom nama_m,tgl_lhr_m, 'Nama Bulan Lahir' dari table Mahasiswa dimana mahasiswa yang tempat lahirnya 'Jakartar', maka perintah nya adalah:

```
SELECT      nama_m,      tgl_lhr_m,"Nama      Bulan
Lahir"=DATENAME(mm,tgl_lhr_m)
FROM Mahasiswa
Where tpt_lhr_m='Jakarta'
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 4 (empat) , seperti tampak pada gambar berikut:

	nama_m	tgl_lhr_m	Nama Bulan Lahir
1	Deni Hermawan	1980-01-17 00:00:00.000	January
2	Muhammad Fahrurozi	1980-10-13 00:00:00.000	October
3	Susilo Wahono	1979-12-21 00:00:00.000	December
4	Yuni Nurwati	1978-11-14 00:00:00.000	November

Gambar 10.59. Query Fungsi DateName

5. Fungsi GetDate

Fungsi ini mempunyai bentuk GetDate(), merupakan fungsi yang digunakan untuk mengetahui tanggal dan waktu sekarang.

Tampilkan kolom 'Tanggal Hari Ini' dan '15 Hari Dari Sekarang', maka perintah nya adalah:

```
SELECT "Tanggal Hari Ini"=GetDate(),"15 Hari Dari
Sekarang"=Getdate()+5
```

Hasil dari pernyataan *query* tersebut adalah 2 (kolom) kolom dan *record* / barisnya adalah 1 (satu) , seperti tampak pada gambar berikut:

	Tanggal Hari Ini	15 Hari Dari Sekarang
1	2006-06-07 13:54:13.203	2006-06-12 13:54:13.203

Gambar 10.60. Query Fungsi GateDate

i. Distinct

Key Word Distinct menghilangkan duplikasi pada baris baris sebagai Default, Query menampilkan semua baris termasuk duplikasinya. Baris-baris yang memiliki duplikasi dapat dieleminasi dengan menggunakan key word Distinct dalam pernyataan Select .

Sintaks dalam pernyataan ini adalah:

```
Select  [ All | Distinct  ]  nama_kolom
From  nama_tabel
Where Kondisi
```

i.1. Untuk 1 (satu) *table*

Tampilkan kolom kota_m dari table Mahasiswa, dimana kolom kota_m diganti menjadi 'Kota Alamat Mahasiswa' dengan Distinct, maka perintah nya adalah:

```
SELECT Distinct kota_m 'Kota Alamat Mahasiswa'
FROM Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah 1 (satu) kolom dan *record* / barisnya adalah 9 (sembilan), seperti tampak pada gambar berikut:



	Kota Alamat Mahasiswa
1	Bekasi Barat
2	Bekasi Timur
3	Bekasi Utara
4	Bogor
5	Cibinong
6	Cibitung
7	Cikarang
8	Jakarta Timur
9	Tambun

Gambar 10.62. Query operator Distinct untuk 1 table

j. Top

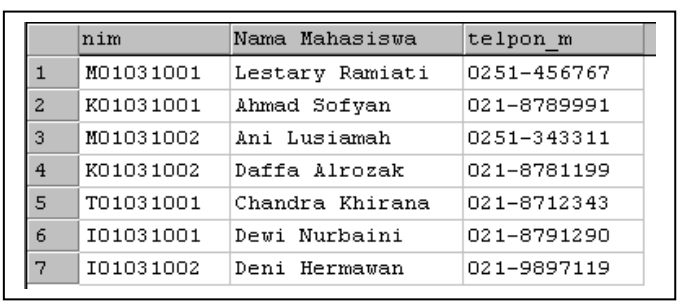
Predikat ini digunakan untuk mengambil sejumlah record yang berada pada jangkauan atas atau bawah dari seluruh data yang diperoleh. Argumen numerik yang diberikan pada predikat Top defaulnya dalam bentuk jumlah record pada jangkauan atas yang ditampilkan. Sedang untuk mengambul n percent dari selumlah record yang didapat, gunakan argumen Percent.

j.1. Untuk 1 (satu) *table*

Tampilkan kolom nim,nama_m,telpon_m dari table Mahasiswa, dimana kolom kota_m diganti menjadi 'Kota Alamat Mahasiswa' dengan data yang ditampilkan 7 record pertama, maka perintah nya adalah:

```
SELECT TOP 7 nim,nama_m 'Nama Mahasiswa',telpon_m
FROM Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 7 (tujuh), seperti tampak pada gambar berikut:



	nim	Nama Mahasiswa	telpon_m
1	M01031001	Lestary Ramiati	0251-456767
2	K01031001	Ahmad Sofyan	021-8789991
3	M01031002	Ani Lusiamah	0251-343311
4	K01031002	Daffa Alrozak	021-8781199
5	T01031001	Chandra Khirana	021-8712343
6	I01031001	Dewi Nurbaini	021-8791290
7	I01031002	Deni Hermawan	021-9897119

Gambar 10.63. Query operator Top untuk 1 table

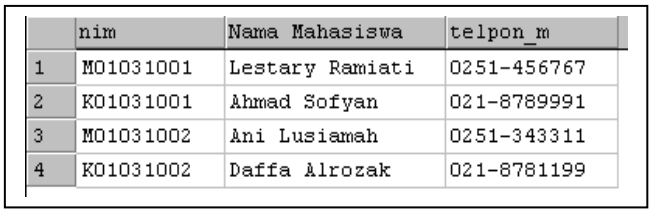
Bila anda ingin mengambil data dalam bentuk prosentasi dari seluruh data yang akan ditampilkan, gunakan argumen Percent setelah argumen numerik.

j.2. Percent untuk 1 (satu) *table*

Tampilkan kolom nim,nama_m,telpon_m dari table Mahasiswa, dimana kolom kota_m diganti menjadi 'Kota Alamat Mahasiswa' dengan data yang ditampilkan 25 % dari seluruh data yang ada, maka perintah nya adalah:

```
SELECT TOP 25 PERCENT nim,nama_m 'Nama Mahasiswa',
      telpon_m
FROM Mahasiswa
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 4 (empat), seperti tampak pada gambar berikut:



	nim	Nama Mahasiswa	telpon_m
1	M01031001	Lestary Ramiati	0251-456767
2	K01031001	Ahmad Sofyan	021-8789991
3	M01031002	Ani Lusiamah	0251-343311
4	K01031002	Daffa Alrozak	021-8781199

Gambar 10.64. Query operator Top Percent untuk 1 table

k. Operator Join

SQL memiliki metode untuk mengambil data dari suatu tabel dengan menggunakan Join. Join dipakai untuk mencari data dari beberapa tabel berdasarkan hubungan yang logis dari tabel-tabel tersebut. Join menyatakan cara SQL memakai data dari sebuah tabel untuk memilih data dari tabel lain. Sintaks untuk melakukan operasi join ini adalah:

Select *nama_kolom,nama_kolom,[nama_kolom]*
From *nama_tabel*, [**Cross / Inner / [Left / Right] Outer]**
Join *nama_tabel* **On** *nama_tabel.Ref_nama_kolom*
OperatorJoin *Nama_tabel.Ref_nama_kolom*.

Keterangan :

Nama_kolom : menspesidikasikan nama kolom dari satu atau beberapa tabel yang ditampilkan.

Nama_tabel : adalah nama tabel dari tabel yang diambil

Ref.nama_kolom : adalah nama_kolom yang digunakan menggabungkan dua tabel dengan menggunakan kunci yang umum. Operator Join menspesifikasikan operator yang digunakan untuk menggabungkan tabel.

Join dapat dikelompokkan menjadi *Inner Join* dan*Outer Join*.

k.1. Inner Join

Inner Join adalah sebuah Join yang menghasilkan baris-baris, minimal ada sebuah baris di kedua tabel yang sesuai dengan kondisi Join. Baris-baris yang tidak sesuai dengan sebuah baris dari tabel lain tidak ditampilkan, defaultnya adalah Inner Join.

a. 3 (tiga) table

Tampilkan nama_m (dari table mahasiswa),nama_mk,semester (dari table Matakuliah),nilai_uts,nilai_uas(dari table Nilai), dimana mahasiswa yang kode_jur ='MI' dan nilainya pada semester=1, maka perintahnya adalah:

```
SELECT nama_m,nama_mk,semester,nilai_uts, nilai_uas
FROM Nilai INNER JOIN Mahasiswa ON
Nilai.nim=Mahasiswa.nim
INNER JOIN Matakuliah ON Nilai.kdmk=Matakuliah.kdmk
Where semester=1 and kode_jur='MI'
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 7 (tujuh), seperti tampak pada gambar berikut:

	nama_m	nama_mk	semester	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN AGAMA I	1	89	55
2	Lestary Ramiati	PENDIDIKAN PANCASILA	1	87	76
3	Ani Lusiamah	PENDIDIKAN AGAMA I	1	78	87
4	Ani Lusiamah	PENDIDIKAN PANCASILA	1	90	90

Gambar 10.65. Query operator inner join 3 table

b. 4 (empat) table

Tampilkan nama_m (dari table mahasiswa),nama_mk,semester (dari table Matakuliah),nilai_uts,nilai_uas(dari table Nilai), nama_d (dari table Dosen), dimana

mahasiswa yang kode_jur ='SI' atau 'TI' dan nilainya pada semester=1, maka perintahnya adalah:

```
SELECT M.nama_m,MK.nama_mk,MK.semester, N.nilai_uts,
N.nilai_uas,D.nama_d
FROM Nilai N INNER JOIN Mahasiswa M ON N.nim=M.nim
INNER JOIN Matakuliah MK ON N.kdmk=MK.kdmk
INNER JOIN Dosen D ON N.nid=D.nid
Where semester=1 and kode_jur in('SI','TI')
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 7 (tujuh), seperti tampak pada gambar berikut:

	nama_m	nama_mk	s...	nilai_uts	nilai_uas	nama_d
1	Dewi Nurbaini	PENDIDIKAN AGAMA I	1	78	89	Susilo Wibowo, M.Kom
2	Dewi Nurbaini	PENDIDIKAN PANCASILA	1	67	78	Sri Mulyani, Ir
3	Rini Hapsari	PENDIDIKAN AGAMA I	1	56	67	Susilo Wibowo, M.Kom
4	Rini Hapsari	PENDIDIKAN PANCASILA	1	45	78	Sri Mulyani, Ir
5	Ira Sulistyaningsih	PENDIDIKAN AGAMA I	1	54	78	Susilo Wibowo, M.Kom
6	Ira Sulistyaningsih	PENDIDIKAN PANCASILA	1	67	88	Sri Mulyani, Ir

Gambar 10.66. Query operator inner join 4 table

k.2. Outer Join

Outer Join akan menghasilkan semua data dari sebuah tabel dan membatasi data dari tabel lain. SQL mempunyai tiga buah tipe Outer Join yaitu Left, Right dan Full. Semua baris dari tabel sebelah kiri diacu oleh sebuah left outer join, dan sebuah baris dari tabel kanan diacu dengan Right outer join. Kegunaan dari outer join adalah mencari record-record piatu, artinya record yang ada disebuah tabel tetapi tidak ada pasangannya pada pada tabel lain.

a. Left Outer Join

Tampilkan nama_m (dari table mahasiswa),nama_mk,semester (dari table Matakuliah),nilai_uts,nilai_uas(dari table Nilai), dimana mahasiswa yang kode_jur ='MI' dan nilainya pada semester=1, maka perintahnya adalah:

```
SELECT nama_mk,semester,sks,nilai_uts,nilai_uas
FROM Matakuliah LEFT OUTER JOIN Nilai ON
Matakuliah.kdmk=Nilai.kdmk
Where semester=2
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 13 (tigabelas), seperti tampak pada gambar berikut:

	nama_mk	semester	sks	nilai_uts	nilai_uas
1	PERANCANGAN BASIS DATA	2	4	NULL	NULL
2	PAKET PROGRAM APLIKASI II	2	1	NULL	NULL
3	ALGORITMA PEMROGRAMAN I (PASCAL)	2	3	NULL	NULL
4	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	2	1	NULL	NULL
5	DASAR MANAJEMEN DAN BISNIS II	2	2	NULL	NULL
6	MATEMATIKA DISKRIT II	2	2	NULL	NULL
7	AKUTANSI DASAR II	2	2	NULL	NULL
8	PENDIDIKAN KEWARGANEGARAAN	2	2	78	89
9	PENDIDIKAN KEWARGANEGARAAN	2	2	66	89
10	PENDIDIKAN KEWARGANEGARAAN	2	2	78	67
11	PENDIDIKAN KEWARGANEGARAAN	2	2	90	76
12	PENDIDIKAN KEWARGANEGARAAN	2	2	57	80
13	BAHASA INGGRIS II	2	2	NULL	NULL

Gambar 10.67. Query operator left outer join

b. Right Outer Join

Tampilkan nama_mk,semester,sks (dari table Matakuliah),nilai_uts,nilai_uas(dari table Nilai), dimana sks matakuliahnya tidak sama dengan 2 (<>2), maka perintahnya asalah:

```
SELECT    MK.nama_mk,MK.semester,MK.sks,N.nilai_uts,
N.nilai_uas
FROM Nilai N  RIGHT OUTER JOIN Matakuliah MK ON
N.kdmk=MK.kdmk
Where sks<>2
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 7 (tujuh), seperti tampak pada gambar berikut:

	nama_mk	semester	sks	nilai_uts	nilai_uas
1	PENGANTAR TEKNOLOGI INFORMASI	1	3	NULL	NULL
2	PRAKTIKUM PAKET PROGRAM APLIKASI I	1	1	NULL	NULL
3	PERANCANGAN BASIS DATA	2	4	NULL	NULL
4	PAKET PROGRAM APLIKASI II	2	1	NULL	NULL
5	ALGORITMA PEMROGRAMAN I (PASCAL)	2	3	NULL	NULL
6	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	2	1	NULL	NULL

Gambar 10.68. Query operator right outer join

c. Full Outer Join

Tampilkan nama_mk,semester,sks (dari table Matakuliah),nilai_uts,nilai_uas(dari table Nilai), dimana matakuliah yang berada pada semester 2, maka perintahnya adalah:

```
SELECT                    MK.nama_mk,MK.semester,MK.sks,
N.nilai_uts,N.nilai_uas
FROM  Nilai  N    FULL  OUTER  JOIN  Matakuliah  MK  ON
N.kdmk=MK.kdmk
Where semester=2
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 13 (tigabelas), seperti tampak pada gambar berikut:

	nama_mk	semester	sks	nilai_uts	nilai_uas
1	PERANCANGAN BASIS DATA	2	4	NULL	NULL
2	PAKET PROGRAM APLIKASI II	2	1	NULL	NULL
3	ALGORITMA PEMROGRAMAN I (PASCAL)	2	3	NULL	NULL
4	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	2	1	NULL	NULL
5	DASAR MANAJEMEN DAN BISNIS II	2	2	NULL	NULL
6	MATEMATIKA DISKRIT II	2	2	NULL	NULL
7	AKUTANSI DASAR II	2	2	NULL	NULL
8	PENDIDIKAN KEWARGANEGARAAN	2	2	78	89
9	PENDIDIKAN KEWARGANEGARAAN	2	2	66	89
10	PENDIDIKAN KEWARGANEGARAAN	2	2	78	67
11	PENDIDIKAN KEWARGANEGARAAN	2	2	90	76
12	PENDIDIKAN KEWARGANEGARAAN	2	2	57	80
13	BAHASA INGGRIS II	2	2	NULL	NULL

Gambar 10.68. Query operator right outer join

i. Pengelompokan Data

Pemakaian perintah select selain dapat digabungkan dengan anak kalimat ORDER BY yang bertujuan untuk mengurutkan data, juga dapat digabungkan dengan anak kalimat GROUP BY yang bertujuan untuk mengelompokkan data (record) yang sama. Pada kasus sederhana anak kalimat GROUP BY berfungsi seperti perintah DISTINCT, kelebihan GROUP BY adalah dapat digunakan bersama dengan fungsi agregate yang beroperasi pada seluruh record yang akan digabung untuk membentuk satu record (baris) tunggal.

a. Order By

Tampilkan nim,nama_m (dari table mahasiswa),nama_jur,jenjang (dari table Jurusan), dimana nama jurusannya bukan 'Teknik Komputer dan diurutkan berdasarkan nama mahasiswa secara Descending, maka perintahnya adalah:

```
SELECT nim,nama_m,nama_jur,jenjang
FROM    Mahasiswa    INNER    JOIN    Jurusan    ON
Mahasiswa.kode_jur=Jurusan.kode_jur
WHERE nama_jur <>'Teknik Komputer'
ORDER BY nama_m DESC
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 13 (tigabelas), seperti tampak pada gambar berikut:

	nim	nama_m	nama_jur	jenjang
1	S01031002	Yudi Ridwan	Sistem Informasi	Strata 1
2	I01031006	Yonita Veronika	Teknik Informatika	Strata 1
3	I01031004	Sylvia Dwita Ningrum	Teknik Informatika	Strata 1
4	I01031005	Susilo Wahono	Teknik Informatika	Strata 1
5	I01031003	Rini Hapsari	Teknik Informatika	Strata 1
6	M01031001	Lestary Ramiati	Manajemen Informatika	Diploma 3
7	S01031001	Ira Sulistyaningsih	Sistem Informasi	Strata 1
8	I01031001	Dewi Nurbaini	Teknik Informatika	Strata 1
9	I01031002	Deni Hermawan	Teknik Informatika	Strata 1
10	K01031002	Daffa Alrozak	Komputerisasi Akuntansi	Diploma 3
11	M01031002	Ani Lusiamah	Manajemen Informatika	Diploma 3
12	K01031001	Ahmad Sofyan	Komputerisasi Akuntansi	Diploma 3

Gambar 10.69. Query operator Order By

b. Group By

b.1. Tampilkan nama_jur (dari table Jurusan), jumlahkan tiap jurusan dan ditampung pada kolom 'Jumlah Mahasiswa' dan dikelompokkan berdasarkan nama jurusan serta diurutkan berdasarkan nama jurusan secara Ascending, maka perintahnya adalah:

```
SELECT      nama_jur      'Jurusan',count(nim)'Jumlah
Mahasiswa'
FROM    Mahasiswa    INNER    JOIN    Jurusan    ON
Mahasiswa.kode_jur=Jurusan.kode_jur
GROUP BY nama_jur
ORDER BY nama_jur
```

Hasil dari pernyataan *query* tersebut adalah 5 (lima) kolom dan *record* / barisnya adalah 13 (tigabelas), seperti tampak pada gambar berikut:

	Jurusan	Jumlah Mahasiswa
1	Komputerisasi Akuntansi	2
2	Manajemen Informatika	2
3	Sistem Informasi	2
4	Teknik Informatika	6
5	Teknik Komputer	4

Gambar 10.70. Query operator Group By (a)

b.2. Tampilkan semester, jumlahkan tiap semester jumlah sksnya ditampung pada kolom 'Jumlah SKS' dan dikelompokkan berdasarkan semester serta diurutkan berdasarkan nama semeser secara Descending, maka perintahnya adalah:

```
SELECT semester,"Jumlah SKS"=sum(sks)
FROM Nilai INNER JOIN Mahasiswa ON
Nilai.nim=Mahasiswa.nim
INNER JOIN Matakuliah ON Nilai.kdmk=Matakuliah.kdmk
GROUP BY Semester
ORDER BY Semester DESC
```

Hasil dari pernyataan *query* tersebut adalah 2 (dua) kolom dan *record* / barisnya adalah 2 (dua), seperti tampak pada gambar berikut:

	semester	Jumlah SKS
1	2	10
2	1	28

Gambar 10.71. Query operator Group By (b)

c. Compute By

c.1. Tampilkan nama_m (dari table Mahasiswa),nama_mk,semester,sks (dari table Jurusana), jumlahkan tiap semester jumlah sksnya dan diurutkan berdasarkan semester, maka perintahnya adalah:

```
SELECT M.nama_m,MK.nama_mk,MK.semester,MK.sks,
N.nilai_uts,N.nilai_uas
FROM Nilai N INNER JOIN Mahasiswa M ON N.nim=M.nim
INNER JOIN Matakuliah MK ON N.kdmk=MK.kdmk
ORDER BY Semester
Compute SUM(sks)BY Semester
```

Hasil dari pernyataan *query* tersebut adalah 6 (enam) kolom, *record* / barisnya adalah 14 (empatbelas), jumlah sksnya = 28 untuk semester 1 dan 5 record dan 10 sks untuk semester 2, seperti tampak pada gambar berikut:

	nama_m	nama_mk	semester	sks	nilai_uts	nilai_uas
1	Lestary Ramiati	PENDIDIKAN AGAMA I	1	2	89	55
2	Lestary Ramiati	PENDIDIKAN PANCASILA	1	2	87	76
3	Ani Lusiamah	PENDIDIKAN AGAMA I	1	2	78	87
4	Ani Lusiamah	PENDIDIKAN PANCASILA	1	2	90	90
5	Daffa Alrozak	PENDIDIKAN AGAMA I	1	2	56	56
6	Daffa Alrozak	PENDIDIKAN PANCASILA	1	2	78	78
7	Chandra Khirana	PENDIDIKAN AGAMA I	1	2	98	89
8	Chandra Khirana	PENDIDIKAN PANCASILA	1	2	88	78
9	Dewi Nurbaini	PENDIDIKAN AGAMA I	1	2	78	89
10	Dewi Nurbaini	PENDIDIKAN PANCASILA	1	2	67	78
11	Rini Hapsari	PENDIDIKAN AGAMA I	1	2	56	67
12	Rini Hapsari	PENDIDIKAN PANCASILA	1	2	45	78
13	Ira Sulistyaningsih	PENDIDIKAN AGAMA I	1	2	54	78
14	Ira Sulistyaningsih	PENDIDIKAN PANCASILA	1	2	67	88

	sum
1	28

	nama_m	nama_mk	semester	sks	nilai_uts	nilai_uas
1	Ira Sulistyaningsih	PENDIDIKAN KEWARGANEGARAAN	2	2	57	80
2	Rini Hapsari	PENDIDIKAN KEWARGANEGARAAN	2	2	78	89
3	Daffa Alrozak	PENDIDIKAN KEWARGANEGARAAN	2	2	66	89
4	Ani Lusiamah	PENDIDIKAN KEWARGANEGARAAN	2	2	90	76
5	Lestary Ramiati	PENDIDIKAN KEWARGANEGARAAN	2	2	78	67

	sum
1	10

Gambar 10.72. Query operator Computer By

j. Sub Query

Sub query adalah sebuah pernyataan select di dalam sebuah Select, Insert, Up date atau Delete atau di dalam subquery lain. Itu semua dapat digunakan untuk mengembalikan data dari beberapa tabel dan sebagai alternatif untuk join. Sub query dapat bersarang atau berhubungan. Sub query yang bersarang dijalankan sekali pada saat Outer Query dijalankan, dan sub query yang berhubungan dijalankan sekali untuk setiap baris dihasilkan pada saat outer query dijalankan.

Berikut ini adalah petunjuk didalam penulisan suatu sub query didalam melakukan penelusuran data didalam database.

- Sub query ditulis dalam tanda kurung.
- Anda hanya boleh memakai sebuah ekspresi atau nama kolom pada daftar select dari sebuah sub query yang menghasilkan sebuah nilai.
- Anda dpat memakai sub query sebagai ganti sebuah ekspresi jika hasilnya adalah sebuah nilai tunggal atau daftar nilai.
- Tidak boleh memakai sub query pada kolom-kolom yang berisi tipe data Teks dan Image.

Sub query dapat dibagi dalam tiga kategori yang bergantung pada nilai pengembalian.

- Where <*ekspresi*> [Not] In (< Subquery>)

Hasil sub query yang memakai IN atau Not IN adalah daftar nilai yang berisi beberapa nilai atau kosong, setelah sub query memberikan hasil, outer query akan memakainya.

a. IN

Contoh berikut ini akan ditampilkan data matakuliah yang pernah dipakai untuk transaksi nilai dengan primary key kdmk, maka perintahnya adalah:

```
SELECT *
FROM Matakuliah M
Where M.kdmk IN
(SELECT N.kdmk From Nilai N)
```

Hasil dari pernyataan *query* tersebut adalah 4 (empat) kolom dan *record* / barisnya adalah 3 (tiga), seperti tampak pada gambar berikut:

	kdmk	nama_mk	sks	semester
1	MPK131201	PENDIDIKAN AGAMA I	2	1
2	MPK131203	PENDIDIKAN PANCASILA	2	1
3	MPK131204	PENDIDIKAN KEWARGANEGARAAN	2	2

Gambar 10.73. Query operator Sub Query IN

b. NOT IN

Penggunaan NOT IN dapat digunakan untuk memeriksa referential integrity antara table relasi dengan table master. Contoh berikut ini akan ditampilkan data matakuliah yang belum pernah dipakai untuk transaksi nilai dan jumlahnya sksnya, maka perintahnya adalah:

```
SELECT      kdmk      'Kode      Matakuliah',nama_mk
'Matakuliah',sks
FROM Matakuliah M
Where sks<>2 and M.kdmk NOT IN
(SELECT N.kdmk From Nilai N)
COMPUTE sum(sks)
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 5 (enam) dan jumlah sksnya = 13, seperti tampak pada gambar berikut:

	Kode Matakuliah	Matakuliah	sks
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3
2	MKB331203	PRAKTIKUM PAKET PROGRAM APLIKASI I	1
3	MKB331205	PERANCANGAN BASIS DATA	4
4	MKB331206	PAKET PROGRAM APLIKASI II	1
5	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3
6	MKB331210	PRAK. ALGORITMA PEMROGRAMAN I (PASCAL)	1

	sum
1	13

Gambar 10.74. Query operator Sub Query NOT IN

c. ANY dan ALL

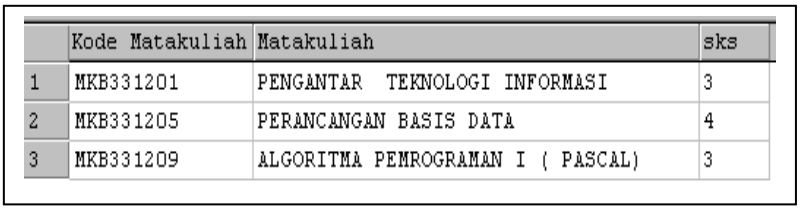
Kata ANY dan ALL dapat digunakan pada subquery – subquery yang menghasilkan satu kolom angka – angka. Jika diawali kata ALL, syarat hanya akan bernilai TRUE jika dipenuhi semua nilai yang dihasilkan subquery ini. Jika diawali kata ANY, syarat akan bernilai TRUE jika dipenuhi sedikitnya satu nilai yang dihasilkan subquery itu.

1. ANY

Perintah berikut adalah menampilkan kolom kdmk, nama_mk,sks dari table matakuliah, untuk kolom kdmk diganti 'Kode Matakuliah', kolom nama_mk diganti 'Matakuliah', dimana hasil subquery adalah sama dengan sks lebih besar 2, maka perintahnya adalah:

```
SELECT      kdmk      'Kode      Matakuliah',nama_mk
'Matakuliah',sks
FROM Matakuliah
Where sks = ANY
(SELECT sks From Matakuliah
WHERE sks>2)
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 3 (tiga), seperti tampak pada gambar berikut:



	Kode Matakuliah	Matakuliah	sks
1	MKB331201	PENGANTAR TEKNOLOGI INFORMASI	3
2	MKB331205	PERANCANGAN BASIS DATA	4
3	MKB331209	ALGORITMA PEMROGRAMAN I (PASCAL)	3

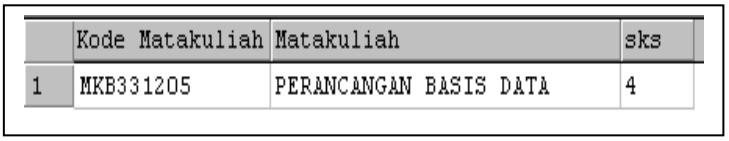
Gambar 10.75. Query operator Sub Query ANY

2. ALL

Perintah berikut adalah menampilkan kolom kdmk, nama_mk,sks dari table matakuliah, untuk kolom kdmk diganti 'Kode Matakuliah', kolom nama_mk diganti 'Matakuliah', dimana hasil subquery adalah sama dengan 3, maka perintahnya adalah:

```
SELECT      kdmk      'Kode      Matakuliah',nama_mk
'Matakuliah',sks
FROM Matakuliah
Where sks > ALL
(SELECT sks From Matakuliah
WHERE sks=3)
```

Hasil dari pernyataan *query* tersebut adalah 3 (tiga) kolom dan *record* / barisnya adalah 3 (tiga), seperti tampak pada gambar berikut:



	Kode Matakuliah	Matakuliah	sks
1	MKB331205	PERANCANGAN BASIS DATA	4

Gambar 10.76. Query operator Sub Query ALL

10.5. SQL - DCL

DBMS menyedikan mekanisme untuk menjamin hanya pemakai yang berhak (diotorisasi) yang dapat mengakses basis data. SQL menyediakan pernyataan GRANT dan REVOKE untuk memungkinkan pengamanan diberikan ke tabel – tabel pada basis data. Pengamanan diberikan pada konsep berikut ini:

1. Identifier Otorisasi (authorization)
2. Kepemilikan (ownership)

3. Wewenang (privelege)

10.5.1. Identifier Otorisasi dan Kepemilikan

Identifier otorisasi adalah identifier SQL normal yang digunakan untuk mengidentifikasi pemakai. Tiap pemakai didalam basis data diberi identifier otorisasi oleh DBA (database administrator) dan biasanya diasosiasikan dengan Pasword. Setiap pernyataan SQL yang dieksekusi oleh DBMS dilakukan spesifik pemakai. Identifier otorisasi digunakan untuk menentukan objek – objek basis data yang boleh diacu pemakai dan operasi – operasi apa yang dapat dilakukan terhadap objek – objek itu.

Tiap objek yang dibuat SQL mempunyai pemilik, pemilik diidentifikasi identifier otorisasi yang didefiniskan klausa AUTHORIZATION dari skema database. Pemilik awalnya merupakan satu – satunya orang yang mengetahui keberadaan objek dan melakukan operasi – operasi terhadap objek itu.

10.5.2. Wewenang

Wewenang adalah aksi – aksi yang boleh dilakukan oleh pemakai pada table dan view. Wewenang yang didefiniskan standard ISO antara lain:

- SELECT, wewenang untuk pengambilan data dari table.
- INSERT, wewenang untuk menyisipkan baris data baru kedalam table.
- UPDATE, wewenang untuk melakukan modifikasi baris data baru di table.
- DELETE, wewenang untuk menghapus baris data didalam table.
- REFERENCES, wewenang untuk mengacu kolom – kolom taable secara terintegrasi.
- USAGE, wewenang menggunakan domain, collation, himpunan karakter dan translation.

10.5.3. Perintah DCL

Perintah - perintah di DCL bergantung dari vendor RDBMS, pada pokoknya hanya terdapat 2 (dua) perintah DCL yaitu GRNT dan REVOKE.

- GRANT

  Pernyataan GRANT digunakan untuk memberikan wewenang pada objek – objek basis data untuk pemakai spesifik. Normalnya, pernyataan GRANT digunakan oleh pemilik table untuk memberikan wewenang kepada pemakai lain untuk mengakses data dengan wewenang tertentu oleh pemilik table.

  Format penyataan GRANT adalah:

GRANT {daftar_wewenang | ALL PRIVELEGES}
ON nama_objek
TO {daftar_id_otorisasi | PUBLIC}
[WITH GRANT OPTION]

Keyword WITH GRANT OPTION memungkinkan pemakai penerima wewenang meneruskan wewenang yang dimilikinya kepada pemakai lainnya.

Jika objek basis data dalah table atau view, maka priveleges dapat berupa:

SELECT
DELETE
INSERT [(column-comma-list)]
UPDATE [(column-comma-list)]
REFERENCES [(column-comma-list)]

Empat option pertama menyatakan GRANT untuk melakukan kalimat – kalimat tertentu. Option berisi option [(column-comma-list)] untuk menyatakan kolom – kolom tertentu.

REFERENCES memberikan GRANT wewenang untuk mengacu table atau kolom menggunakan Foreign Key. Tampaknya seperti aneh untuk mengendalikan pengaksesan ini, namun demikianlah. Pengaksesan foreign key yang tidak terbatas dapat membuka lubang keamanan.

- REVOKE

Pernyataan REVOKE digunakan untuk mencabut wewenang pada objek – objek basis data dan dapat digunakan untuk mencabut semua wewenang atau beberapa wewenang pemakai.

Format penyataan GRANT adalah:

REVOKE [GRANT OPTION WHERE] {daftar_wewenang | ALL PRIVELEGES}
ON nama_objek
FROM {daftar_id_otorisasi | PUBLIC} [RESRICT | CASCADE]

CASCADE berarti jika suatu pemakai yang mempunyai otorisasi telah memberikan wewenang itu kepada pemakai lain, maka wewenang yang telah diberikannya juga turut dicabut.

Demikian juga, jika pemakai kedua telah memberikan wewenang kepada pemakai berikutnya, maka wewenang pemakai berikutnya inipun ikut dicabut, demikian seterusnya.

RESTRICT berarti jika wewenang bergantung masih ada, maka kalimat REVOKE dibatalkan.