BINA INSANI UNIVERSITY

# SQL
# (Structured Query Language)

## Pertemuan 10

Bigger Better Higher

# RDBMS

- RDBMS adalah program 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

# DEFINISI *S Q L*

- *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.

# SEJARAH *S Q L*

- 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".

# BAGIAN BAHASA S Q L

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.

# BAGIAN BAHASA S Q L...

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.

# SUB DIVISI S Q L

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 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.

# SUB DIVISI S Q L ...

c. DCL (Data Control Language)

DCL berisi fitur – fitur 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:

(1) View definiton, SQL DDL untuk perintah mendifinisikan View.

(2) Transaction control, untuk menspesifikasikan awal an akhir transaksi dan melakukan pengendalian transaksi.

# SUB DIVISI S Q L ...

(3) 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.

(4) Integrity, perintah untuk menspesifikasikan konstrain – konstrain integritas dimana data disimpan di basisdata yang harus dipenuhi DBMS. Pembaruan yang melanggar konstrain – kontrain integritas ditolak.

# 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.

# SQL – 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:
  *a)CREATE* object_name
  *b)ALTER* object_name
  *c)DROP* object_name

# SQL – DDL …

- 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.

# 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:
  1) Identifier tidak boleh lebih panjang dari 148 karakter (kebanykaan dialek lebih pendek).
  2) Identifier harus dimulai dengan huruf.
  3) Identifier tidak boleh berisi spasi.

# 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.
- 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<br>SMALLINT |
| 4. | Numerik riil | FLOAT | REAL | DOUBLE PRICISION |
| 5. | Waktu tanggal | DATE | TIME | |
| 6. | interval | INTERVAL | | |

# Tipe Data MS SQL Server

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])<br>Varbinary[(n]) | -<br>binary varying[(n]) | 1-8000 |
| Character | Char[(n)]<br>Varchar[[n)] | Character[(n)]<br>char[acter]varying[(n)] | 1-8000<br>(8000 characters) |
| Unicode character | Nchar[(n)]<br>Nvarchar[(n)] | National char[acter][(n)]<br>National char[acter]varying[(n)] | 2-8000<br>(1-4000 characters) |
| Date and time | Datetime, smalldatetime | - | 8 (24 byte integers)<br>4 (22 byte integers) |

Lanjutan.....

| Tipe data | Tipe data disediakan system | Sinonim dengan ANSI | Jumlah byte |
|---|---|---|---|
| Exact numeric | Decimal[(p[,s])]<br>numeric[(p[,s])] | dec | 5-17 |
| Approximate numeric | Float[(n)]<br>Real | Double precision or Float[(n)] | 4-8<br>4 |
| Global identifier | Uniqueidentifier | - | 16 |
| Integer | Int<br>smallint, tinyint | Integer<br>- | 4<br>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 |

- 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 yang didefinisikan pemakai.
2) sp_droptype, untuk menghapus tipe data yag didefinisikan pemakai.

## 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
```

# Pendefinisian 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.

# Pendefinisian Basis Data...

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.*

*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”
```

# 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]}
```

Keterangan:

`table_name` :adalah nama tabel yang akan dibuat.

`column_name` :adalah nama-nama atribut yang akan terdapat di dalam tabel_name.

# Pembuatan Table...

- 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.

# Struktur Table Mahasiswa

| | 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

## 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)
)
```

# Struktur Table Mahasiswa

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.

# Membuat Table langsung dibentuk Primary Key-nya

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)
)
```

# 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)
```

# Merubah Struktur Table...

## 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
```

# Penghapusan Table

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

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

```
Delete * From Mahasiswa
```

Atau :

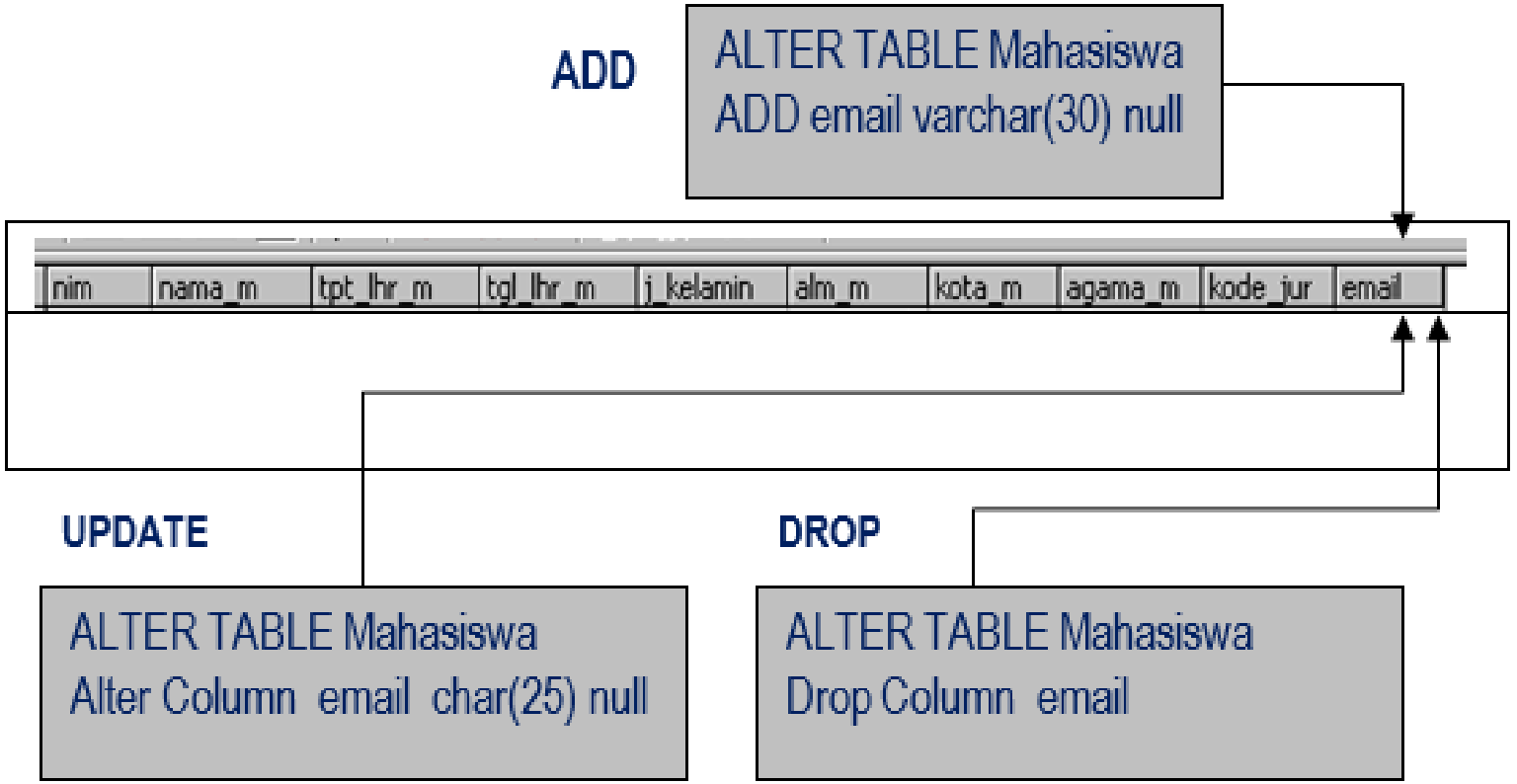
```
Delete from Mahasiswa
```

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

```
Drop Table Mahasiswa
```

# Manipulasi Terhadap Definisi Kolom

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



# Manipulasi Terhadap Definisi Kolom…..

✓**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.

# 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 lain.
- 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 – DML ….

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.*

# SQL – DML ….

- 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.

# SQL – DML ….

- 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.


Thank you