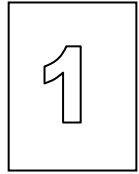


PENGANTAR BASIS DATA



Obyektif :

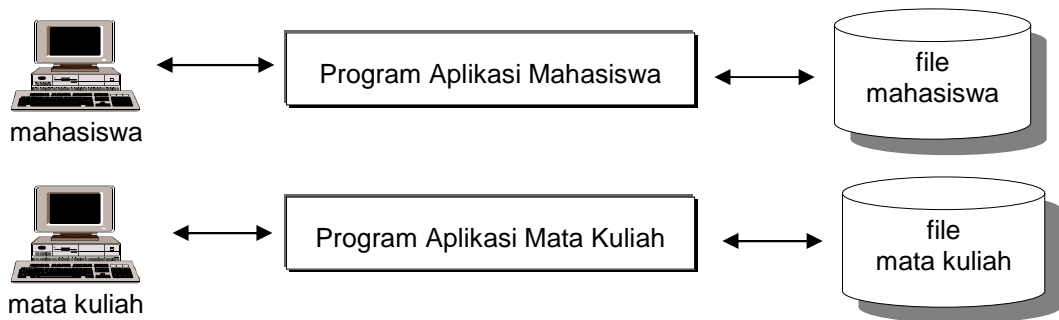
1. Menjelaskan perbedaan antara file tradisional dan file manajemen basis data
 2. Menjelaskan keuntungan dan kerugian apabila menggunakan file manajemen basis data
 3. Memahami konsep basis data dan istilah yang termasuk didalamnya
 4. Mengetahui bahasa yang digunakan untuk pengoperasian basis data
-

Basis data menyediakan fasilitas atau mempermudah dalam menghasilkan informasi yang digunakan oleh pemakai untuk mendukung pengambilan keputusan. Hal inilah yang menjadikan alasan dari penggunaan teknologi basis data pada saat sekarang (dunia bisnis).

Berikut ini contoh penggunaan Aplikasi basis data dalam dunia bisnis :

- Bank : Pengelolaan data nasabah, akunting, semua transaksi perbankan
- Bandara : Pengelolaan data reservasi, penjadualan
- Universitas : Pengelolaan pendaftaran, alumni
- Penjualan : Pengelolaan data customer, produk, penjualan
- Pabrik : Pengelolaan data produksi, persediaan barang, pemesanan, agen
- Kepegawaian: Pengelolaan data karyawan, gaji, pajak
- Telekomunikasi : Pengelolaan data tagihan, jumlah pulsa

Sistem Pemrosesan File



Gambar 1. Sistem pemrosesan file untuk suatu Universitas

Keterangan :

File mahasiswa : **Mhs** (npm, nama, alamat, tgl_lahir)

MataKul (kd_mk, nama_mk, sks)

File MataKuliah : **MataKul** (kd_mk, nama, sks)

Sebelumnya, sistem yang digunakan untuk mengatasi semua permasalahan bisnis, menggunakan pengelolaan data secara tradisional dengan cara menyimpan record-record pada file-file yang terpisah, yang disebut juga sistem pemrosesan file. Dimana masing-masing file diperuntukkan hanya untuk satu program aplikasi saja. Perhatikan gambar 1 mengenai suatu universitas yang mempunyai dua sistem yakni sistem yang memproses data mahasiswa dan sistem yang mengelola data mata kuliah.

Kelemahannya dari sistem pemrosesan file ini antara lain :

1. Timbulnya data rangkap (*redundancy data*) dan Ketidakkonsistensi data (*Inconsistency data*)

Karena file-file dan program aplikasi disusun oleh programmer yang berbeda, sejumlah informasi mungkin memiliki duplikasi dalam beberapa file. Sebagai contoh nama mata kuliah dan sks dari

mahasiswa dapat muncul pada suatu file memiliki record-record mahasiswa dan juga pada suatu file yang terdiri dari record-record mata kuliah. Kerangkapan data seperti ini dapat menyebabkan pemborosan tempat penyimpanan dan biaya akses yang bertambah. Disamping itu dapat terjadi inkonsistensi data. Misalnya, apabila terjadi perubahan jumlah sks mata kuliah, sedangkan perubahan hanya diperbaiki pada file mata kuliah dan tidak diperbaiki pada file mahasiswa. Hal ini dapat mengakibatkan kesalahan dalam laporan nilai mahasiswa.

2. Kesukaran dalam Mengakses Data

Munculnya permintaan-permintaan baru yang tidak diantisipasi sewaktu membuat program aplikasi, sehingga tidak memungkinkan untuk pengambilan data.

3. Data terisolir (*Isolation Data*)

Karena data tersebar dalam berbagai file, dan file-file mungkin dalam format –format yang berbeda, akan sulit menuliskan program aplikasi baru untuk mengambil data yang sesuai.

4. Masalah Pengamanan (*Security Problem*)

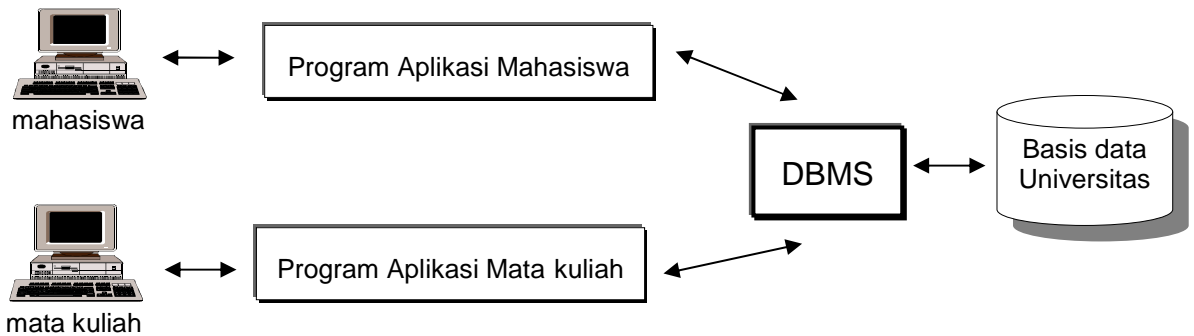
Tidak semua pemakai diperbolehkan mengakses seluruh data. Bagian Mahasiswa hanya boleh mengakses file mahasiswa. Bagian Mata kuliah hanya boleh mengakses file mata kuliah, tidak boleh mengakses file mahasiswa. Tetapi sejak program-program aplikasi ditambahkan secara ad-hoc maka sulit melaksanakan pengamanan seperti yang diharapkan.

5. Data Dependence

Apabila terjadi perubahan atau kesalahan pada program aplikasi maka pemakai tidak dapat mengakses data.

Sistem Basis data

Seiring dengan berjalannya waktu, lambat laun sistem pemrosesan file mulai ditinggalkan karena masih bersifat manual, yang kemudian dikembangkanlah sistem pemrosesan dengan pendekatan basis data.



Gambar 2. Sistem basis data untuk suatu universitas

Keterangan :

Mhs (Npm, nama, alamat, tgl_lahir)

Mt_kul (kd_mk, nama_mk,sks)

Perhatikan gambar 2 di atas. Pada sistem ini record-record data disimpan pada satu tempat yakni basis data dan diantara program aplikasi maupun pemakai terdapat DBMS (*Database Management System*).

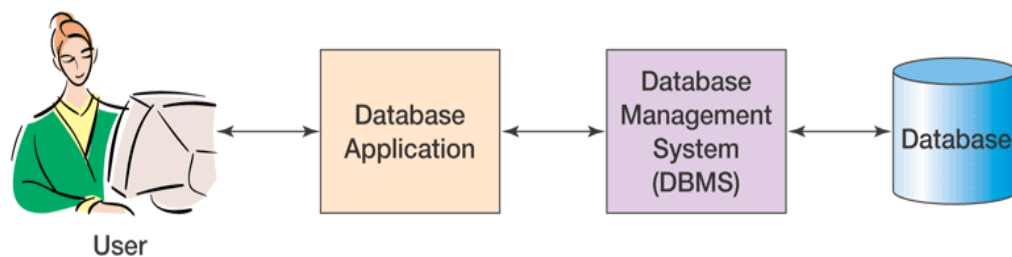
Konsep Dasar Basis Data

Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, mahasiswa, pembeli), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.

Basis Data adalah sekumpulan data yang terintegrasi yang diorganisasikan untuk memenuhi kebutuhan para pemakai di dalam suatu organisasi.

DBMS (Database Management System) adalah Perangkat Lunak yang menangani semua pengaksesan ke basis data

Sistem Basis Data terdiri dari basis data dan DBMS.



Gambar 3. Sistem Basis Data

Istilah - Istilah Dasar Basis Data

Enterprise

Suatu bentuk organisasi seperti : bank, universitas, rumah sakit, pabrik, dsb.

Data yang disimpan dalam basis data merupakan data operasional dari suatu enterprise.

Contoh data operasional : data keuangan, data mahasiswa, data pasien

Entitas

Suatu obyek yang dapat dibedakan dari lainnya yang dapat diwujudkan dalam basis data.

Contoh Entitas dalam lingkungan bank terdiri dari : Nasabah, Simpanan, Hipotik

Contoh Entitas dalam lingkungan universitas terdiri dari : Mahasiswa, mata kuliah

Kumpulan dari entitas disebut **Himpunan Entitas**

Contoh : semua nasabah, semua mahasiswa

Atribut (Elemen Data)

Karakteristik dari suatu entitas.

Contoh : Entitas Mahasiswa atributnya terdiri dari Npm, Nama, Alamat, Tanggal lahir.

Nilai Data (Data Value)

Isi data / informasi yang tercakup dalam setiap elemen data.

Contoh Atribut Nama Mahasiswa dapat berisi Nilai Data : Diana, Sulaeman, Lina

Kunci Elemen Data (*Key Data Element*)

Tanda pengenal yang secara unik mengidentifikasi entitas dari suatu kumpulan entitas.

Contoh Entitas Mahasiswa yang mempunyai atribut-atribut npm, nama, alamat, tanggal lahir menggunakan Kunci Elemen Data npm.

Record Data

Kumpulan Isi Elemen data yang saling berhubungan.

Contoh : kumpulan atribut npm, nama, alamat, tanggal lahir dari Entitas Mahasiswa berisikan : "10200123", "Sulaeman", "Jl. Sirsak 28 Jakarta", "8 Maret 1983".

Keuntungan Sistem Basis Data

1. Terkontrolnya kerangkapan data

Dalam basis data hanya mencantumkan satu kali saja field yang sama yang dapat dipakai oleh semua aplikasi yang memerlukannya.

2. Terpeliharanya keselarasan (kekonsistenan) data

Apabila ada perubahan data pada aplikasi yang berbeda maka secara otomatis perubahan itu berlaku untuk keseluruhan

3. Data dapat dipakai secara bersama (*shared*)
Data dapat dipakai secara bersama-sama oleh beberapa program aplikasi (secara *batch* maupun *on-line*) pada saat bersamaan.
4. Dapat diterapkan standarisasi
Dengan adanya pengontrolan yang terpusat maka DBA dapat menerapkan standarisasi data yang disimpan sehingga memudahkan pemakaian, pengiriman maupun pertukaran data.
5. Keamanan data terjamin
DBA dapat memberikan batasan-batasan pengaksesan data, misalnya dengan memberikan password dan pemberian hak akses bagi pemakai (misal : *modify, delete, insert, retrieve*)
6. Terpeliharanya integritas data
Jika kerangkapan data dikontrol dan kekonsistenan data dapat dijaga maka data menjadi akurat
7. Terpeliharanya keseimbangan (keselarasan) antara kebutuhan data yang berbeda dalam setiap aplikasi
Struktur basis data diatur sedemikian rupa sehingga dapat melayani pengaksesan data dengan cepat
8. *Data independence* (kemandirian data)
Dapat digunakan untuk bermacam-macam program aplikasi tanpa harus merubah format data yang sudah ada

Kelemahan Sistem Basis Data

- Memerlukan tenaga spesialis
- Kompleks
- Memerlukan tempat yang besar
- Mahal

Pengguna Basis Data

1. *System Engineer*

Tenaga ahli yang bertanggung jawab atas pemasangan Sistem Basis Data, dan juga mengadakan peningkatan dan melaporkan kesalahan dari sistem tersebut kepada pihak penjual

2. *Database Administrator (DBA)*

Tenaga ahli yang mempunyai tugas untuk mengontrol sistem basis data secara keseluruhan, meramalkan kebutuhan akan sistem basis data, merencanakannya dan mengaturnya.

Tugas DBA :

- Mengontrol DBMS dan *software-software*
- Memonitor siapa yang mengakses basis data
- Mengatur pemakaian basis data
- Memeriksa *security, integrity, recovery* dan *concurrency*

Program Utilitas yang digunakan oleh DBA :

- *Loading Routines*
Membangun versi utama dari basis data
- *Reorganization Routines*
Mengatur / mengorganisasikan kembali basis data
- *Journaling Routines*
Mencatat semua operasi pemakaian basis data
- *Recovery Routines*
Menempatkan kembali data, sebelum terjadinya kerusakan
- *Statistical Analysis Routines*
Membantu memonitor kehandalan sistem

3. *End User (Pemakai Akhir)*

Ada beberapa jenis (tipe) pemakai terhadap suatu sistem basis data yang dapat dibedakan berdasarkan cara mereka berinteraksi terhadap sistem :

a. Programmer aplikasi

Pemakai yang berinteraksi dengan basis data melalui *Data Manipulation Language* (DML), yang disertakan (*embedded*) dalam program yang ditulis pada bahasa pemrograman induk (seperti C, pascal, cobol, dll)

b. Pemakai Mahir (*Casual User*)

Pemakai yang berinteraksi dengan sistem tanpa menulis modul program. Mereka menyatakan *query* (untuk akses data) dengan bahasa *query* yang telah disediakan oleh suatu DBMS

c. Pemakai Umum (*End User / Naïve User*)

Pemakai yang berinteraksi dengan sistem basis data melalui pemanggilan satu program aplikasi permanen (*executable program*) yang telah ditulis (disediakan) sebelumnya

d. Pemakai Khusus (*Specialized/Sophisticated User*)

Pemakai yang menulis aplikasi basis data non konvensional, tetapi untuk keperluan-keperluan khusus seperti aplikasi AI, Sistem Pakar, Pengolahan Citra, dll, yang bisa saja mengakses basis data dengan atau tanpa DBMS yang bersangkutan.

Structure Query Language

Structure Query Language (SQL) merupakan komponen bahasa *relational database system*. SQL merupakan bahasa baku (ANSI/SQL), *non procedural*, dan berorientasi himpunan (*set-oriented language*). SQL dapat digunakan baik secara interaktif atau ditempelkan (*embedded*) pada sebuah program aplikasi.

Komponen-Komponen SQL

a. **Data Definition Language (DDL) :**

Digunakan untuk mendefinisikan data dengan menggunakan perintah : *create, drop, alter*.

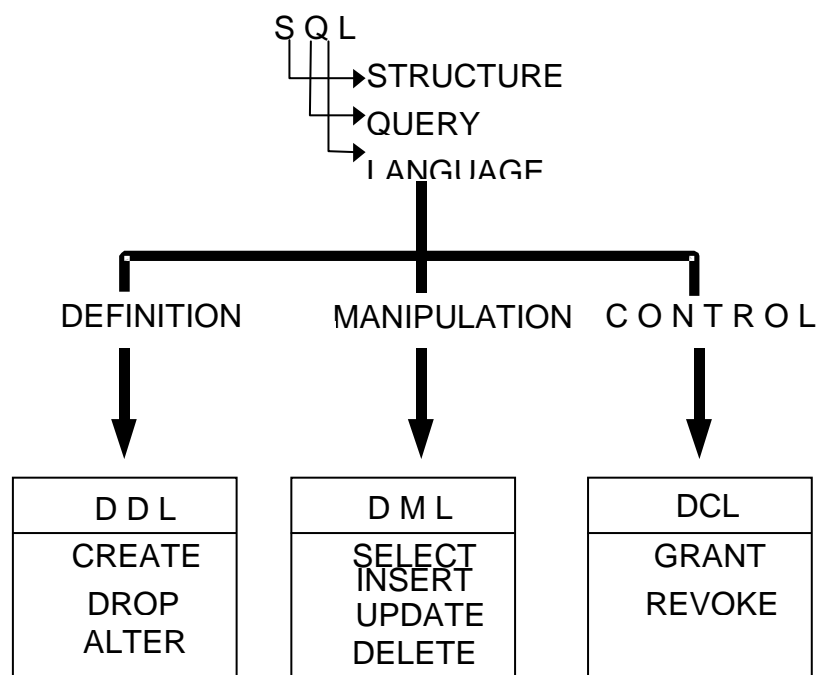
b. **Data Manipulation Language (DML) :**

Digunakan untuk memanipulasi data dengan menggunakan perintah : *select, insert, update, delete.*

Data Manipulation Language merupakan bagian terpadu bahasa SQL. Perintah-perintahnya dapat dibuat secara interaktif atau ditempelkan pada sebuah program aplikasi. Pemakai hanya perlu menentukan 'APA' yang ia inginkan, DBMS menentukan 'BAGAIMANA' cara mendapatkannya.

c. **Data Control Language (DCL) :**

Digunakan untuk mengontrol hak para pemakai data dengan perintah : *grant, revoke*



DATA DEFINITION LANGUAGE : CREATE, DROP, ALTER

2

Obyektif :

5. Mengetahui dan memahami perintah yang terdapat pada Data Definition Language
 6. Dapat menggunakan perintah CREATE, DROP, dan ALTER
-

1. CREATE TABLE

Fungsi : membuat tabel

Sintaks : **CREATE TABLE tname**

```
(col 1      data type  data spec,  
 col 2      data type  data spec,  
 .  
 .  
 PRIMARY KEY (col1,.....))
```

Contoh :

```
CREATE TABLE PERSONEL  
(REGNO  CHAR(10)  NOT NULL,  
 NAME   CHAR(45)  NOT NULL,  
 ADDRESS CHAR(45),  
 BIRTH  DATE  NOT NULL WITH DEFAULT,  
 PRIMARY KEY (REGNO))
```

NULL

Spesifikasi *NULL, NOT NULL, NOT NULL WITH DEFAULT*

NULL :

dapat diinterpretasikan sebagai nilai yang tidak diketahui atau tidak tersedianya suatu nilai. Null bukan berarti kosong (blank) atau 0 (Nol)

NOT NULL :

pemakai atau program harus memberikan nilai-nilai pada saat memasukkan record

NOT NULL WITH DEFAULT :

nilai *default* disimpan pada saat record dimasukkan tanpa nilai yang ditentukan untuk kolom ini.

Nilai *default*-nya :

Nol	untuk tipe field NUMERIC
Blank	untuk tipe field CHARACTER
CURRENT DATE	untuk tipe field DATE
CURRENT TIME	untuk tipe field TIME

Pada saat membuat tabel, salah satu atribut tersebut di atas dispesifikasikan pada sebuah kolom.

2. CREATE VIEW

Fungsi : membuat tabel view.

View merupakan bentuk alternatif penyajian data dari satu atau lebih tabel. View dapat berisi semua atau sebagian kolom yang terdapat pada tabel dimana kolom tersebut didefinisikan.

Tujuan membuat view :

- Meningkatkan keamanan data
- Meningkatkan kemandirian data
- Penyederhanaan bagi end user (data yang sedikit, nama-nama kolom yang baru dan dapat dibaca dengan lebih baik)

Properti :

- Tidak terdapatnya data tambahan
- View mencakup subset kolom dan / atau baris
- View dapat berisikan data dari beberapa tabel dan / atau tabel-tabel view lainnya
- View dapat berisikan perolehan data, misal : nilai rata-rata
- Manipulasi data melalui view terbatas

Sintaks : **CREATE VIEW viewname (column1, column2,)
AS SELECT statement FROM tname
[WITH CHECK OPTION]**

Keterangan :

View-name : nama view yang akan dibuat.

Column : nama atribut untuk view

Statement : atribut yang dipilih dari tabel basis data.

Tabel-name : nama tabel basis data.

Contoh :

```
CREATE VIEW VPERSON (REGNO, NAME) AS  
SELECT REGNO, NAME FROM PAUL.PERSONEL
```

3. CREATE INDEX

Fungsi : membuat index

Sintaks : **CREATE [UNIQUE] INDEX indexname**
ON nama_table (nama_kolom)

Contoh :

```
CREATE UNIQUE INDEX PRSONIDX  
ON PERSONEL(REGNO)
```

Dengan indeks memungkinkan suatu tabel diakses dengan urutan tertentu tanpa harus merubah urutan fisik dari datanya dan dapat pula diakses secara cepat melalui indeks yang dibuat berdasar nilai field tertentu. Spesifikasi UNIQUE akan menolak key yang sama dalam file.

4. DROP TABLE

Fungsi : menghapus Tabel

Sintaks : **DROP TABLE tname**

Contoh : DROP TABLE PERSONEL

Dengan perintah itu obyek lain yang berhubungan dengan tabel tersebut otomatis akan dihapus atau tidak akan berfungsi seperti :

- semua record dalam tabel akan terhapus
- index dan view pada tabel akan hilang
- deskripsi tabel akan hilang

5. DROP VIEW

Fungsi : menghapus view

Sintaks : **DROP VIEW viewname**

Contoh : DROP VIEW VPERSON

6. DROP INDEX

Fungsi : menghapus index

Sintaks : **DROP INDEX indexname**

Contoh : DROP INDEX PRSONIDX

7. ALTER

Fungsi : merubah atribut pada suatu tabel

Sintaks : **ALTER TABLE tname**

MODIFY (nama_kolom tipe_kolom)

**ADD (nama_kolom tipe_kolom [[before,
nama_kolom]])**

DROP (nama_kolom tipe_kolom)

Contoh : merubah Tabel TABX dengan menambah Field D.

ALTER TABLE TABX

ADD D CHAR(3)

Contoh Kasus DDL :

- **Membuat Tabel (CREATE TABLE)**

1. CREATE TABLE S

```
(Sn      Char(5)  NOT NULL,  
 Sname Char(20)  NOT NULL,  
 Status Smallint NOT NULL,  
 City   Char(15) NOT NULL);
```

2. CREATE TABLE P

```
(Pn      Char(6)  NOT NULL,  
 Pname Char(20)  NOT NULL,  
 Color  Char(6)  NOT NULL,  
 Weight Smallint NOT NULL);
```

3. CREATE TABLE SP

```
(Sn Char(5)      NOT NULL,  
 Pn Char(6)      NOT NULL,  
 QTY INTEGER     NOT NULL);
```

4. CREATE UNIQUE INDEX Sidx ON S(Sn);
CREATE UNIQUE INDEX Pidx ON P(Pn);
CREATE INDEX Sdx ON SP(Sn);
CREATE INDEX Pdx ON SP(Pn);

- **Modifikasi Table P dengan perintah :**

```
RENAME COLUMN P.COLOR TO WARNA  
ALTER TABLE P ADD (City CHAR(15) NOT NULL)
```


- **Membuat View (CREATE VIEW)**

1. Membuat view untuk supplier yang statusnya lebih besar dari 15

```
CREATE VIEW GOOD_SUPPLIERS
AS SELECT Sn, Status, City FROM S
WHERE Status > 15;
```

2. Membuat view yang berisi supplier yang tinggal di Paris

```
CREATE VIEW Paris_Suppliers
AS SELECT * FROM Suppliers
WHERE City = ' Paris '
```

3. Membuat view dengan mengganti nama_atributnya

```
CREATE VIEW Parts (PNum, Part_Name, WT)
AS SELECT P#, Pname, Weight FROM Part
WHERE COLOR = 'Red'
```

DATA MANIPULATION LANGUAGE : INSERT, SELECT, UPDATE, DELETE

3

Obyektif :

7. Mengetahui dan memahami perintah yang terdapat pada Data Manipulation Language
 8. Dapat menggunakan perintah INSERT, SELECT, UPDATE, dan DELETE
-

1. INSERT

FUNGSI : MENAMBAH BARIS (RECORD) BARU

Sintaks : **INSERT INTO tname**
(col1, ...) VALUES (value1, ...)

Catatan :

Sintaks tersebut dapat digunakan jika jumlah kolom = jumlah nilai, tetapi jika dalam tabel semua kolom akan diisi dapat digunakan sintaks berikut ini :

Sintaks : **INSERT INTO tname**
VALUES (value1, value2, ...)

Nilai-nilai diisikan sebanyak kolom yang terdapat di tabel tersebut.

2. UPDATE

Fungsi : merubah record

Sintaks : **UPDATE** tname
SET field = ekspresi
WHERE kondisi

3. DELETE

Fungsi : menghapus record

Sintaks : **DELETE FROM** tname

WHERE kondisi

4. SELECT

Fungsi : menampilkan record

Sintaks : **SELECT** [DISTINCT] colname FROM tname
[WHERE kondisi]
[GROUP BY kondisi]
[HAVING kondisi]
[ORDER BY kondisi]

Contoh Kasus DML :

- **Menambah record (INSERT)**

```
INSERT INTO S VALUES ('S1','Smith',20,'London');
```

```
INSERT INTO S VALUES ('S2','Jones',10,'Paris');
```

```
INSERT INTO S VALUES ('S3','Blake',30,'Paris')
```

Tabel S, P dan SP isikan dengan data-data sebagai berikut :

TABEL S

Sn	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

TABEL P

Pn	Pname	Warna	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

TABEL SP

Sn	Pn	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- **Merubah record (UPDATE)**

1. Merubah data (record) pada tabel P yang mempunyai nomor part P2, warnanya dirubah menjadi Kuning dan beratnya ditambah 5

UPDATE P

```
SET Warna = 'Yellow',  
    Weight = Weight + 5  
WHERE Pn = 'P2'
```

2. Merubah record pada tabel S, statusnya menjadi dua kali status awal untuk supplier yang bertempat tinggal di kota London

```
UPDATE S
    SET Status = 2 * Status
    WHERE City = 'London'
```

- **Menghapus record (DELETE)**

Menghapus record pada tabel S yang nomor supplier-nya S5

```
DELETE FROM S
    WHERE Sn = 'S5'
```

- **Menampilkan record (SELECT 1 tabel)**

1. Menampilkan semua data supplier

```
SELECT * FROM S
atau
SELECT Sn, Sname, Status, City FROM S
```

2. Menampilkan semua nilai Pn pada tabel SP

```
SELECT Pn FROM SP
```

3. Menampilkan nomor supplier dan status untuk supplier yang tinggal di Paris

```
SELECT Sn, Status FROM S
    WHERE City = 'Paris'
```

4. Menampilkan no.supplier yang tinggal di Paris dengan status > 20

```
SELECT Sn FROM S
      WHERE City ='Paris" AND Status > 20
```

5. Menampilkan jumlah pengiriman P1

```
SELECT COUNT(*) FROM SP
      WHERE Pn = 'P1'
```

6. Perintah untuk menghindari hasil data yang sama terulang kembali (distinct)

```
SELECT DISTINCT Pn FROM SP
```

7. Menampilkan no.supplier dan status bagi supplier yang tinggal di Paris dalam urutan status menurun

```
SELECT Sn,Status FROM S
      WHERE City = 'Paris'
      ORDER BY Status desc
```

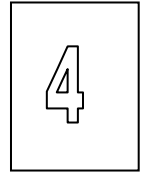
8. Menampilkan no.Part dari semua part yang dipasok oleh lebih dari seorang supplier

```
SELECT Pn FROM SP
      GROUP BY Pn
      HAVING COUNT(*) > 1
```

9. Menampilkan semua part yang nomornya dimulai dengan huruf C

```
SELECT * FROM P  
WHERE Pname LIKE 'C%'
```


DATA MANIPULATION LANGUAGE : AGGREGATE FUNCTION, GROUP BY, HAVING



Obyektif :

9. Mengetahui dan memahami perintah Fungsi Agregasi, Group By, dan Having
 10. Dapat menggunakan perintah Fungsi Agregasi, Group By, dan Having
-

Selain mengambil data dalam basis data dengan kriteria tertentu, sering juga diperlukan berbagai perhitungan yang bersifat ringkasan. Fungsi AGREGASI digunakan untuk melakukan operasi pada kelompok-kelompok baris data sehingga menghasilkan satu baris data untuk setiap kelompok baris data yang ada.

Fungsi Agregasi adalah berikut ini :

Fungsi	Deskripsi
AVG	Menghitung rata-rata nilai ekspresi yang tidak NULL. B.U: SELECT AVG(kolom) FROM tabel
SUM	Menghitung total nilai ekspresi yang tidak NULL. B.U: SELECT SUM(kolom) FROM tabel
COUNT	Menghitung banyaknya baris yang dipilih secara query. B.U: SELECT COUNT(kolom) FROM tabel

	Mencari nilai maksimal dari suatu kolom.
MAX	B.U: SELECT MAX(kolom) FROM tabel
	Mencari nilai minimal dari suatu kolom.
MIN	B.U: SELECT MIN(kolom) FROM tabel
	Menampilkan nilai dari record yang pertama dari suatu kolom/field.
FIRST	B.U: SELECT FIRST(kolom) AS [ekspresi] FROM tabel
	Menampilkan nilai dari record terakhir dari suatu kolom/field.
LAST	B.U: SELECT LAST(kolom) AS [ekspresi] FROM tabel

Misalnya untuk menghitung jumlah pegawai, rata-rata gaji, dan total gaji pegawai dari tabel Pegawai. Maka perintahnya adalah :

```
SELECT  COUNT(*),
        AVG(gaji),
        SUM(gaji)
FROM    Pegawai;
```

- **Fungsi agregasi — GROUP BY**

Digunakan untuk membentuk group/kelompok dari tupel-tupel yang memiliki nilai yang sama. Hasil pengelompokan dirutkan secara ascending.

Misalnya untuk menghitung jumlah pegawai, rata-rata gaji, total gaji untuk setiap kelompok golongan pegawai. Maka sintaks perintahnya adalah :

```
SELECT  COUNT(*),
        AVG(gaji),
        SUM(gaji)
FROM    Pegawai
GROUP BY golongan;
```

- **Fungsi agregasi – HAVING**

Klausa HAVING digunakan untuk membatasi baris yang dihasilkan oleh fungsi agregasi GROUP BY.

Misalnya untuk menghitung rata-rata gaji dan total gaji dalam tabel Pegawai untuk setiap kelompok golongan pegawai yang rata-rata gaji > 1000000. Maka perintahnya adalah :

```
SELECT  AVG(gaji),
        SUM(gaji)
FROM    Pegawai
GROUP BY golongan
HAVING  AVG(gaji)>1000000;
```

- **Fungsi agregasi – ORDER BY**

Digunakan untuk mengurutkan query dengan nilai yang berisi dalam satu atau lebih kolom. Secara *default* data diurutkan secara ascending (menaik).

Contoh :

- (1) Mengurutkan data berdasarkan nama pegawai. Maka sintaks perintahnya adalah:

```
SELECT      *
FROM        Pegawai
ORDER BY    nama;
```

- (2) Mengurutkan data pegawai berdasarkan nama, dan diurutkan lagi berdasarkan golongan secara menurun. Maka sintaks perintahnya adalah :

```
SELECT      *
FROM        Pegawai
ORDER BY    nama,
            golongan DESC;
```

Contoh Kasus :

Tabel PERSON

Nama	Umur
Cynthia	24
Kevin	32
Dave	45
Jenny	20
Michael	24
David	

Dengan menggunakan tabel di atas,

1. Untuk menampilkan umur tertua (maksimal) :

SELECT MAX(umur) FROM PERSON;

Jawab : **45**

2. Untuk menampilkan umur termuda (minimal) :

SELECT MIN(umur) FROM PERSON;

Jawab : **20**

3. Untuk menghitung banyaknya data :

SELECT COUNT(*) FROM PERSON; → data dari seluruh record

Jawab : **6**

SELECT COUNT(umur) FROM PERSON; → data di kolom umur, tidak termasuk yang null

Jawab : **5**

4. Untuk menghitung total umur :

SELECT SUM(umur) FROM PERSON;

Jawab : **135**

5. Untuk mengurutkan data :

SELECT * FROM PERSON ORDER BY nama;

Jawab :

Tabel PERSON

Nama	Umur
Cynthia	24
Dave	45
David	
Jenny	20
Kevin Michael	32
Michael	24

→ Diurutkan secara
menaik (ascending)

SELECT * FROM PERSON ORDER BY nama DESC;
(mengurutkan data berdasarkan nama secara
menurun/descending)

6. Untuk menampilkan record pertama:

**SELECT FIRST(umur) AS umur_termuda
FROM PERSON ORDER BY umur;**

Jawab : **20**

7. Untuk menampilkan record terakhir :

**SELECT LAST(umur) AS umur_tertua
FROM PERSON ORDER BY umur;**

Jawab : **45**

DATA CONTROL LANGUAGE : GRANT DAN REVOKE

5

Obyektif :

11. Mengetahui dan memahami perintah Data Control Language
 12. Dapat menggunakan perintah Grant dan Revoke
-

Data Control Language (DCL) merupakan perintah-perintah yang dapat digunakan untuk menjaga keamanan basis data. Perintah tersebut dapat dipakai untuk menentukan akses basis data hanya dapat dilakukan oleh orang-orang tertentu dan dengan macam akses yang dibatasi pula. Adapun Objek-Objek DCL dalam Mysql diantaranya :

a. Tabel USER dari database MySQL

Tabel user adalah tabel yang ada dalam database MySQL. Tabel user hanya diperuntukkan bagi seorang Administrator (root). Tabel user bersifat global, apapun perubahan yang terjadi pada tabel ini akan mempengaruhi jalannya keseluruhan system MySQL.

Dari kolom yang didapat dari tabel user untuk MySQL yang embeded pada PHP Instant adalah sebagai berikut :

Field	Type	Null	Key	Default	Extra
Host	Varchar(60) binary		Pri		
User	Varchar(16) binary		Pri		
Password	Varchar(16)				
Select_priv	Enum('N','Y')			N	
Insert_priv	Enum('N','Y')			N	
Update_priv	Enum('N','Y')			N	
Delete_priv	Enum('N','Y')			N	
Drop_priv	Enum('N','Y')			N	
Reload_priv	Enum('N','Y')			N	
Shutdown_priv	Enum('N','Y')			N	
Process_priv	Enum('N','Y')			N	
File_priv	Enum('N','Y')			N	
Grant_priv	Enum('N','Y')			N	
References_priv	Enum('N','Y')			N	
Index_priv	Enum('N','Y')			N	
Alter_priv	Enum('N','Y')			N	
Show_db_priv	Enum('N','Y')			N	
Create_priv	Enum('N','Y')			N	

Penjelasan fungsi :

Field	Penjelasan
Host	Field ini berisi alamat host komputer user. Field ini dapat diisi "localhost" jika hanya mengizinkan user yang bersangkutan hanya dapat mengakses dari komputer yang sama dengan server MySQLnya Dapat juga diisi dengan "%" jika user MySQL dapat mengakses dari

	komputer lain dalam jaringan. Jika hanya mengizinkan user dapat mengakses dari satu komputer saja, maka masukkan nilainya dengan nomor IP address pada komputer klien sehingga user yang bersangkutan tidak dapat mengakses server MySQL dari komputer lain termasuk komputer tempat MySQL terinstall, kecuali hanya dari komputer yang nomor IP-nya sesuai dengan yang dimasukkan pada kolom host.
User	Kolom ini berisi username dari user yang terdaftar. Username tersebut digunakan untuk melakukan login pada server MySQL sebagai user biasa. Username dapat berupa pencampuran karakter (huruf, angka, simbol karakter)
Password	Berisi password dari username yang terdaftar Isi dari kolom ini akan di-enkripsi dengan standard password yang dimiliki oleh MySQL server
Select_priv	Berfungsi untuk memberikan hak user untuk menampilkan data (menggunakan perintah SELECT) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah SELECT Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah SELECT
Insert_priv	Berfungsi untuk memberikan hak user untuk memasukkan data (menggunakan perintah INSERT) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah INSERT Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah INSERT
Update_priv	Berfungsi untuk memberikan hak user untuk meremajakan / memperbarui (menggunakan perintah UPDATE) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah UPDATE Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah UPDATE
Delete_priv	Berfungsi untuk memberikan hak user untuk menghapus / mengosongkan data (menggunakan perintah DELETE) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah DELETE Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah DELETE
Drop_priv	Berfungsi untuk memberikan hak user untuk menghapus database, tabel atau kolom (menggunakan perintah DROP) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah DROP Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah DROP
Reload_priv	Berfungsi untuk memberikan hak user untuk melakukan fereshe server (menggunakan perintah RELOAD) Jika diisi 'Y' berarti user diijinkan untuk menggunakan perintah RELOAD Jika diisi 'T' berarti user tidak diijinkan untuk menggunakan perintah RELOAD
Shutdown_priv	Berfungsi untuk memberikan hak user untuk dapat mematikan daemon server MySQL Jika diisi 'Y' berarti user diijinkan untuk menghentikan server Jika diisi 'T' berarti user tidak diijinkan untuk menghentikan server
Process_priv	Berfungsi untuk memberikan hak user untuk dapat melihat proses

- **Sintaks Umum**

1. Menampilkan data pada kolom host, user dan password saja

```
SELECT host, user, password FROM user;
```

2. Menghapus semua user tanpa identitas

```
DELETE FROM user WHERE user='';
```

3. Mengganti password

```
UPDATE user SET password=password('xxxxxxxxxx')  
WHERE user='root';
```

4. Perintah FLUSH.

```
FLUSH PRIVILEGES;
```

Perintah FLUSH PRIVILEGES adalah suatu perintah untuk mengaktifkan perubahan-perubahan yang terjadi pada user, seperti hak akses, penggantian password pada user, dsb. Perintah FLUSH PRIVILEGES ini hukumnya wajib dilaksanakan setelah Anda melakukan perubahan (apapun juga) secara langsung ke dalam tabel user atau ke dalam database mysql.

- **Contoh penggunaan :**

1. Menggunakan database mysql

```
use mysql;
```

2. Menambah user ='Winda' dan password='w1nd'

```
insert into user (User,Password) values('Winda','w1nd');
```

3. Menambah user ='Febe' dan password='4567'

```
insert into user (User,Password) values('Febe','4567');
```

4. Menambah user ='Elfrida' dan password='9812'

```
insert into user (User,Password) values('Elfrida','9812');
```

5. menampilkan user dan password tabel user

```
select user,password from user;
```

6. Menghapus semua user tanpa identitas

```
Delete from user where user='';
```

7. Menampilkan semua isi dari tabel user

```
Select * from user;
```

8. Mengubah password untuk user winda dengan 'w1nd4'

```
Update user set password ='w1nd4' where user='Winda';
```

9. Mengubah password untuk user Febe dengan 'f3b3'

```
Update user set password ='f3b3' where user='Febe';
```

10. Mengubah password untuk user Elfrida dengan 'fr1d4'

```
Update user set password =' fr1d4' where user='Elfrida';
```

b. Tabel Tables_Priv dari database MySQL

Tabel_priv berfungsi mengatur tabel apa saja yang dapat diakses oleh seorang user, berikut jenis izin aksesnya. Tingkat akses hanya untuk tabel. Pada prinsipnya hanya bekerja seperti db table, kalau tidak digunakan untuk tabel sebagai ganti database.

- **Sintaks Umum :**

SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW

- **Contoh penggunaan :**

1. Insert into tables_priv(User,table_name,table_priv)
values('Febe,Elfrida','Point_Of_Sales.jenis','select');
2. Insert into tables_priv(User,table_name,table_priv)
values('Winda','Point_Of_Sales.jenis','select,insert');
3. Insert into tables_priv(User,table_name,table_priv)
values('Elfrida','Point_Of_Sales.item','select,insert,update');
4. Insert into tables_priv (User,Table_name,Table_priv)
values ('Admin','Point_Of_Sales.user , Point_Of_Sales.Jenis,
Point_Of_Sales.item, Point_Of_Sales.JualGlobal,,
Point_Of_Sales.JualDetail', 'SELECT, INSERT, UPDATE, DELETE,
CREATE, DROP, GRANT, REFERENCES, INDEX, ALTER,
CREATE VIEW, SHOW VIEW');
5. Insert into tables_priv (User,Table_name,Table_priv)
values('kasir','Point_Of_Sales.jualDetail','select,insert,show view');

6. Insert into tables_priv (User,Table_name,Table_priv)
values('Cewek','Point_Of_Sales.jenis, Point_Of_Sales.item,
Point_Of_Sales.jualGlobal, Point_Of_Sales.jualDetail',
'select,insert,update,show view');
7. update tables_priv set host='localhost' where user='Admin';
8. update tables_priv set host='192.168.10.2' where user='Cewek';
9. update tables_priv set grantor ='root@localhost' where user='Admin';
- 10.delete from tables_priv where user='cewek';

c. GRANT

Grant berfungsi untuk membuat user baru dan memberikan hak istimewa. Grant adalah salah satu privilege untuk tabel. Grant digunakan untuk memberikan privilege kepada tabel yang didefinisikan kepada pemakai lain. Privilege untuk pemakai dalam perintah grant didefinisikan dengan menggunakan nama-nama privilege. Nama privilege memudahkan administrator untuk dapat memberikan privilege tanpa harus tahu apa nama field dan tabel yang harus diisi.

Perintah grant secara otomatis akan menambah data pemakai apabila data nama pemakai yang disertakan pada perintah tersebut belum ada dalam tabel user. Perintah grant memudahkan administrator untuk tidak perlu melakukan perintah pendefinisian privilege dengan menggunakan sql. Karena dengan menggunakan sql, kita harus hafal nama tabel yang harus diisi, field apa saja yang harus diisi, jumlah field yang harus diisi. Kesalahan mudah dilakukan dengan menggunakan perintah sql karena

ketidaktelitian atau ketidakhafalan nama tabel dan nama field yang harus diisi.

- **Sintak Umum :**

- ✓ GRANT *hak_akses* ON *nama_tabel* TO *pemakai*;
- ✓ GRANT ALL PRIVILEGES ON *database_name.** TO 'myuser'
IDENTIFIED BY 'mypassword';

- **Contoh Penggunaan :**

1. GRANT SELECT ON Point_Of_Sales.jenis TO Febe;
2. GRANT SELECT ON Point_Of_Sales.jenis TO Winda;
3. GRANT SELECT ON Point_Of_Sales.item TO Elfrida;
4. GRANT ALL PRIVILEGES ON Point_Of_Sales.User TO Admin;
5. GRANT ALL ON Point_Of_Sales.jualDetail TO Admin
6. SHOW GRANTS FOR root@localhost;
7. SHOW GRANTS FOR Admin;
8. GRANT SELECT,INSERT ON Point_Of_Sales.jualDetail TO kasir;
9. GRANT SELECT(Kode>Nama) ON Point_Of_Sales.jenis TO Elfrida;
10. GRANT UPDATE(kodeItem,NmlItem,kategori,Harga) ON
Point_Of_Sales.item TO Elfrida;

d. REVOKE

Untuk menghapus batasan hak akses yang telah diatur dengan menggunakan perintah GRANT, digunakan perintah **REVOKE**.

- **Sintak umum :**

- ✓ REVOKE *hak_akses* ON *nama_tabel* FROM
namaAccount@namaHost;

Atau menghapus batasan hak akses untuk database & tabel :

- ✓ REVOKE *hak_akses* ON *nama_database.nama_tabel* FROM *user*;

Atau menghapus batasan hak akses untuk kolom tertentu :

- ✓ REVOKE *hak_akses(field1,field2, field3,...)* ON
nama_database.nama_tabel FROM *user*;

Penulisan perintah REVOKE :

- ✓ **HakAkses(field)** : kita harus memberikan sedikitnya satu hak akses. Untuk setiap hak akses yang diberikan, dapat juga diberikan daftar field yang diletakkan dalam kurung, dan dipisahkan dengan tanda koma. Contoh : REVOKE select (nim, nama), update, insert(nim), ...
- ✓ **NamaTabel** : merupakan nama tabel yang dikenal hak akses tersebut. Harus ada sedikitnya satu nama tabel. Dapat menggunakan simbol asterik (*) untuk mewakili semua tabel pada database aktif. Penulisan namaTabel dapat juga diikuti oleh nama database diikuti nama tabel yang dipisahkan dengan tanda titik. Menggunakan simbol *.* berarti semua database dan semua tabel yang dikenai hak akses tersebut.

✓ **namaAccount@namaHost** : jika nama account tidak ada, tidak pernah diberikan hak akses dengan perintah GRANT sebelumnya maka akan terjadi error.

• **Contoh Penggunaan :**

1. REVOKE SELECT ON Point_Of_Sales.jenis FROM Febe;
2. REVOKE SELECT ON Point_Of_Sales.jenis FROM Winda;
3. REVOKE SELECT ON Point_Of_Sales.item FROM Elfrida;
4. REVOKE ALL PRIVILEGES ON Point_Of_Sales.user FROM Admin;
5. REVOKE ALL ON Point_Of_Sales.jualDetail FROM Admin;
6. REVOKE SELECT,INSERT ON Point_Of_Sales.jualDetail FROM kasir;
7. REVOKE SELECT(Kode>Nama) ON Point_Of_Sales.jenis FROM Elfrida;
8. REVOKE UPDATE(kodeItem,NmlItem,kategori,Harga) ON Point_Of_Sales.item FROM Elfrida;
9. REVOKE INSERT ON Point_Of_Sales.jenis FROM Winda;
10. REVOKE ALL ON Point_Of_Sales.item FROM PUBLIC;

e. SHOW PROCESSLIST

Digunakan untuk menampilkan kegiatan apa saja yang terjadi pada MySQL server atau menampilkan informasi mengenai thread yang dieksekusi di server. Bila terdapat kegiatan yang membahayakan kita sebagai admin dapat menghentikan dengan perintah KILL atau MySQLAdmin

- Sintak umum :

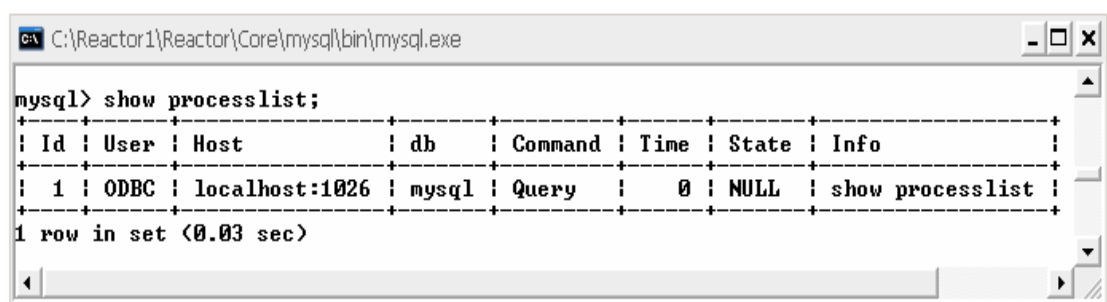
SHOW PROCESSLIST;

Id	User	Host	db	command	time	state	Info
1	ODBC	127.0.0.1:1030	Null	Sleep	6		Null
2	ODBC	127.0.0.1:1031	Pendaftaran	Query	0	Null	Show Processlist

2 rows in set (0.00 set)

- Contoh penggunaan :

SHOW PROCESSLIST;



```
C:\Reactor1\Reactor\Core\mysql\bin\mysql.exe
mysql> show processlist;
+----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host           | db    | Command | Time | State | Info           |
+----+-----+-----+-----+-----+-----+-----+-----+
| 1  | ODBC | localhost:1026 | mysql | Query   | 0    | NULL  | show processlist |
+----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.03 sec)
```

f. KILL

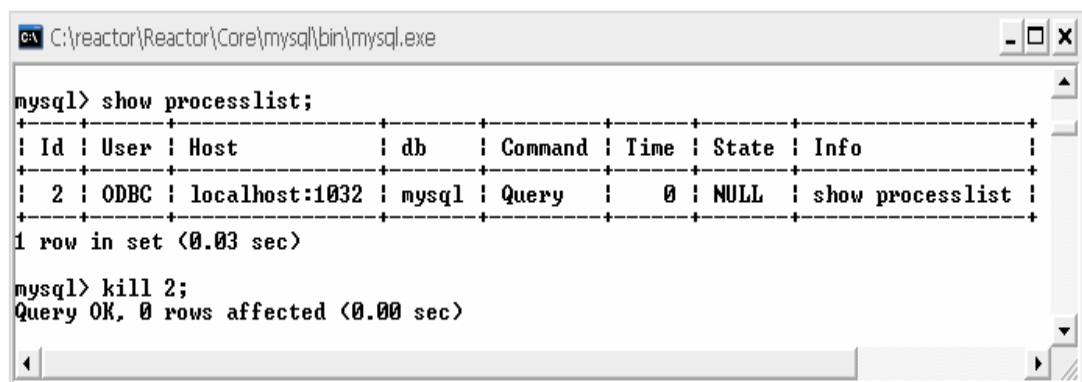
Kill berfungsi menghentikan thread server / untuk membunuh proses yang sedang berjalan

- **Sintak Umum :**

KILL nomor_Id;

- **Contoh penggunaan :**

Kill 2;



```
C:\reactor\Reactor\Core\mysql\bin\mysql.exe
mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host          | db   | Command | Time | State | Info          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2  | ODBC | localhost:1032 | mysql | Query   | 0    | NULL  | show processlist |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.03 sec)

mysql> kill 2;
Query OK, 0 rows affected (0.00 sec)
```

g. OPTIMIZE TABLE

- ✓ Tabel yang sering mengalami proses penghapusan dan penambahan akan menyebabkan struktur yang tidak teratur secara fisik atau telah terjadi fragmentasi.
- ✓ Penghapusan data dalam jumlah besar mempunyai peluang terjadinya fragmentasi.
- ✓ Terutama untuk data bertipe VARCHAR, TEXT atau BLOB

- ✓ Tidak semua DBMS dapat melakukan fragmentasi, kita dapat melihat dukungan setiap DBMS
- ✓ Untuk mengatasi masalah fragmentasi solusinya adalah melakukan OPTIMIZE TABLE
- ✓ Untuk MySQL versi 3.23 keatas mendukung fasilitas OPTIMIZE TABLE.
- ✓ Perlu diketahui pada saat OPTIMIZE TABLE dikerjakan, semua tabel akan di-lock (terkunci)
- ✓ Proses fragmentasi sebaiknya dilakukan secara berkala, misalnya setiap minggu atau setiap bulan.

▪ **Sintak umum :**

OPTIMIZE TABLE *tabel_1, tabel_2, tabel_3, tabel_n;*

Table	Op	Msg_Type	Mst_Text
STMIK.Mhs	Optimize	Status	OK
STMIK.Absen	Optimize	Status	OK
STMIK.Nilai	Optimize	Status	Table is already up to date

6 rows in set (1.37 sec)

• **Contoh penggunaan :**

**OPTIMIZE TABLE Point_Of_Sales.User,Point_Of_Sales.item,
Point_Of_Sales.jualGlobal;**

JOIN

6

Obyektif :

13. Mengetahui dan memahami perintah JOIN

14. Dapat menggunakan perintah JOIN

Perintah JOIN digunakan untuk menampilkan suatu output yang berasal dari beberapa tabel (lebih dari satu tabel).

Contoh :

TABEL S

Sn	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

TABEL P

Pn	Pname	Warna	Weight	City
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

TABEL SP

Sn	Pn	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- **Menampilkan record (SELECT lebih dari satu tabel / JOIN)**

1. Menampilkan semua supplier dan part yang keduanya bertempat tinggal pada kota yang sama

```
SELECT Sn, Sname, S.tatus, S.City , Pn, Pname, Warna, Weight  
FROM S,P  
WHERE S.City = P.City
```

2. Menampilkan nama supplier yang memasok barang dengan nomor part P2

```
SELECT Sname FROM S, SP  
WHERE S.Sn = SP.Sn AND SP.Pn = 'P2'
```

3. Menampilkan nama supplier yang memasok part berwarna merah

```
SELECT Sname FROM S, SP, P
WHERE S.Sn = SP.Sn
      AND SP.Pn = P.Pn
      AND P.COLOR = 'RED'
```

- **Menampilkan record (SELECT lebih dari satu tabel / SELECT Bertingkat)**

1. Menampilkan nama supplier yang memasok barang dengan nomor part P2

```
SELECT Sname FROM S WHERE Sn IN
      (SELECT Sn FROM SP WHERE Pn = 'P2')
```

atau

```
SELECT Sname FROM S WHERE Sn = ANY
      (SELECT Sn FROM SP WHERE Pn = 'P2')
```

2. Menampilkan nama supplier yang memasok part berwarna merah

```
SELECT Sname FROM S WHERE Sn IN
      (SELECT Sn FROM SP WHERE Pn IN
        (SELECT Pn FROM P WHERE Warna = 'Red'))
```

3. Menampilkan no.supplier dengan nilai status lebih kecil daripada nilai maksimum status yang ada pada tabel S

```
SELECT Sn FROM S WHERE Status <
      (SELECT MAX(Status) FROM S)
```

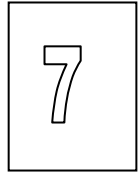
4. Menampilkan nama supplier yang tidak memasok barang dengan nomor part P2

```
SELECT Sname FROM S WHERE Sn NOT IN  
      (SELECT Sn FROM SP WHERE Pn = 'P2')
```

5. Menampilkan semua nomor supplier yang sama lokasinya dengan S1

```
SELECT Sn FROM S WHERE CITY =  
      (SELECT CITY FROM S WHERE Sn = 'S1')
```

SUBQUERY



Obyektif :

15. Mengetahui dan memahami perintah Subquery
 16. Dapat menggunakan perintah Subquery
 17. Memproduksi hasil yang dapat digunakan pada seleksi berikutnya.
-

Subquery (Subselect) adalah pernyataan SELECT yang merupakan bagian dari pernyataan lain, misal : INSERT. Pernyataan ORDER BY, FOR UPDATE OF, UNION, INTERSECT atau EXCEPT tidak termasuk dalam pernyataan ini.

Subquery menghasilkan sebuah tabel yang merupakan bagian dari tabel atau view yang diidentifikasi pada klausa FROM. Pembagian ini dapat digambarkan seperti urutan operasi, dimana hasil dari suatu operasi adalah input bagi operasi lain.

Subquery diperlukan pada saat hasil query tidak berhasil dilakukan dengan hanya melalui satu tabel saja, juga pada saat hasil suatu query digunakan pada klausa WHERE query lainnya. Hasil yang diperoleh dari SUBSELECT tidak dapat ditampilkan oleh "main" SELECT.

Urutan operasi pada *Subquery* adalah :

1. klausa FROM
2. klausa WHERE
3. klausa GROUP BY
4. klausa HAVING
5. klausa SELECT

- **Sintaks :**

```
SELECT    ( * | (ekspresi_kolom) ....  
FROM      (nama_tabel) ....  
[WHERE    kondisi ]  
[GROUP BY (nama_kolom) .... ]  
[HAVING   kondisi_having ]
```

- **SUBQUERY – Coding**

Fungsi :

Pada klausa kondisi (WHERE atau HAVING), akses lain seperti SELECT dapat melibatkan beberapa tabel. Ada beberapa cara untuk menggabungkan SELECT tambahan pada klausa SELECT atau HAVING :

- Perbandingan aritmatik (=, >, <)
- ANY (dikombinasikan dengan =, >=, <=)
- SOME (dikombinasikan dengan =, >=, <=)
- IN

Hasil *Subquery* menentukan key word manakah yang dapat digunakan (ANY, SOME, IN). UNION tidak diperkenankan untuk digunakan dalam SUBSELECT. Sedangkan aritmatik dapat digunakan.

Contoh :

1. *Subquery* dengan Perbandingan Aritmatik

```
SELECT EMPNO, LASTNAME, M_SALARY  
FROM OWNER_ID.EMP  
WHERE M_SALARY >  
      (SELECT AVG(M_SALARY)  
       FROM OWNER_ID.EMP)
```

Subquery menghasilkan nilai tunggal. Oleh karena itu perbandingan dalam klausa WHERE cukup dilakukan dengan operator aritmatik yang sederhana.

2. *Subquery* dengan IN

```
SELECT EMPNO, FIRSTNAME, M_SALARY  
FROM OWNER_ID.EMP  
WHERE M_SALARY > 2500 AND JOBID IN  
      (SELECT JOBID FROM OWNER_ID.JOB  
       WHERE JOB_NAME LIKE 'SYSTEM%')
```

Subquery ini menghasilkan sekumpulan nilai tetapi hasilnya masih dalam satu kolom. Key word khusus dibutuhkan untuk menggabungkan nilai-nilai tersebut dalam “main” SELECT. =ANY atau =SOME dapat digunakan sebagai pengganti IN.