

# PENGELOLAAN BASIS DATA

06. SQL Query Select

07. SQL Query Where / Groups

11. SQL DDL

Doni Abdul Fatah

Universitas Trunojoyo Madura

# Pokok Bahasan (Revisi)

01. Database Design Review

02. Database User Account Management

03. Database Backup & Recovery

04. Database Replication

05. Database Optimization

06. SQL Query Select

07. SQL Query Where / Groups

08. SQL Query Join

09. Stored Procedure / Function

10. Trigger

11. SQL DDL

12. Optimising DB Structure

13. Project

14. UAS

# Penjelasan Pokok Bahasan (Revisi)

Database Design Review	Database User Account Management	Database Backup & Recovery	Database Replication
<ul style="list-style-type: none"><li>• Relational Model</li><li>• ERD</li><li>• Conceptual Database Design</li><li>• Logical Database Design</li><li>• Physical Database Design</li></ul>	<ul style="list-style-type: none"><li>• Grant</li><li>• Revoke</li><li>• Privelege</li><li>• Adding User</li><li>• Limiting User Resource</li><li>• Tugas Besar 1</li></ul>	<ul style="list-style-type: none"><li>• Backup</li><li>• Restore</li><li>• Check</li><li>• Repair</li><li>• Table Maintenance</li></ul>	<ul style="list-style-type: none"><li>• Overview</li><li>• Setup Replication</li><li>• Sql command (slave)</li></ul>

# Penjelasan Pokok Bahasan (Revisi)

Database Optimization	SQL Query Select	SQL Query Where / Groups	SQL Query Join
<ul style="list-style-type: none"><li>• Definisi</li><li>• Design limitation / tradeoff</li><li>• Benchmarking</li><li>• Explain</li><li>• Query performance</li></ul>	<ul style="list-style-type: none"><li>• Select, where</li><li>• Insert, update, delete</li><li>• Join</li><li>• Union</li><li>• Truncate</li><li>• replace</li></ul>	<ul style="list-style-type: none"><li>• Select, where</li><li>• Insert, update, delete</li><li>• Join</li><li>• Union</li><li>• Truncate</li><li>• replace</li></ul>	<ul style="list-style-type: none"><li>• Select, where</li><li>• Insert, update, delete</li><li>• Join</li><li>• Union</li><li>• Truncate</li><li>• replace</li></ul>

# Penjelasan Pokok Bahasan (Revisi)

Stored Procedure / Function	Trigger	SQL DDL	Optimising DB Structure
<ul style="list-style-type: none"><li>• Definisi</li><li>• Parameter</li><li>• Body</li><li>• Local variable</li><li>• Set</li><li>• Flow control</li><li>• Call stored procedure</li><li>• Error message</li></ul>	<ul style="list-style-type: none"><li>• Konsep Trigger</li><li>• Create trigger</li><li>• Trigger as Integrity constraints</li><li>• Trigger and catalog</li></ul>	<ul style="list-style-type: none"><li>• Definisi</li><li>• Creating tables</li><li>• Data types</li><li>• <b>Temporary table</b></li><li>• <b>Copying tables</b></li><li>• Naming tables and columns</li><li>• Table options</li><li>• Integrity constraints</li><li>• Primary keys</li><li>• Alternate keys</li><li>• Foreign keys</li></ul>	<ul style="list-style-type: none"><li>• Konsep Design choices</li><li>• Indexes</li><li>• Multiple column indexes</li><li>• Tuning server parameter</li><li>• How mySQL uses memory</li></ul>

# 01. Pengelolaan Basis Data

- 1) Database User Account Management
- 2) SQL DDL
- 3) SQL Query Select
- 4) SQL Query Where / Groups
- 5) Proyek Akhir
- 6) Referensi

# ENTITY RELATIONSHIP DIAGRAM (ERD)

Model data / tool (alat) yang digunakan dalam proses analisa untuk menggambarkan kebutuhan data dan asumsi-asumsi dalam sistem, secara top-down (dari atas ke bawah). ERD ini dapat digunakan kembali (berulang) untuk analisa dan desain pada SDLC (System Development Life Cycle)

Tiga elemen dasar di dalam ERD:

- **Entities** adalah “sesuatu” dimana kita mencari informasi.
- **Attributes** adalah kumpulan data pada entity.
- **Relationships** adalah penghubung antara entity-entity.

# RDBMS

## a) model Entity-Relationship

- ❑ Entity adalah obyek yang dapat dibedakan dalam dunia nyata
- ❑ Entity set adalah kumpulan dari entity yang sejenis, Entity set dapat berupa :
  - ❑ Obyek secara fisik : Rumah, Kendaraan, Peralatan
  - ❑ Obyek secara konsep : Pekerjaan , Perusahaan, Rencana



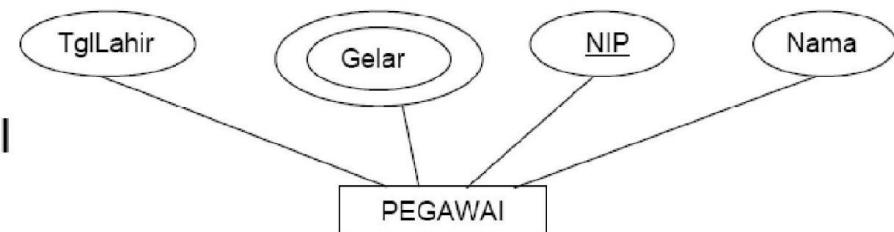
# RDBMS

## a) Atribut Entity-Relationship

- Atribut adalah karakteristik dari entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relationship tersebut.
- Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu entity atau relationship.

Jenis-jenis atribut :

- **Key**  
Atribut yang digunakan untuk menentukan suatu entity secara unik.
- **Atribut Simple**  
Atribut yang bernilai tunggal
- **Atribut Multivalue**  
Atribut yang memiliki sekelompok nilai untuk setiap instan entity.

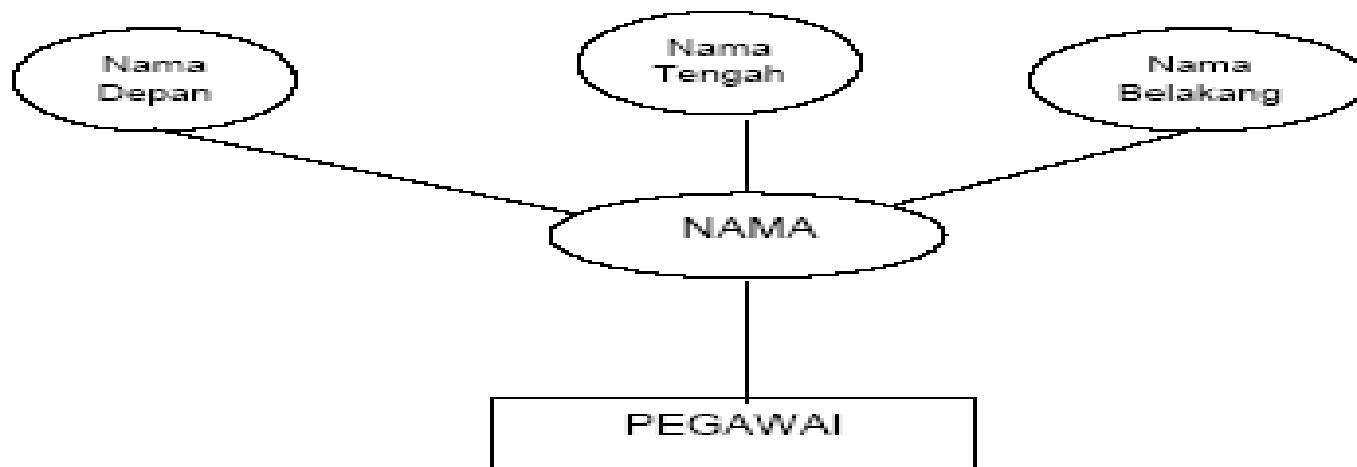


# RDBMS

## a) Atribut Entity-Relationship

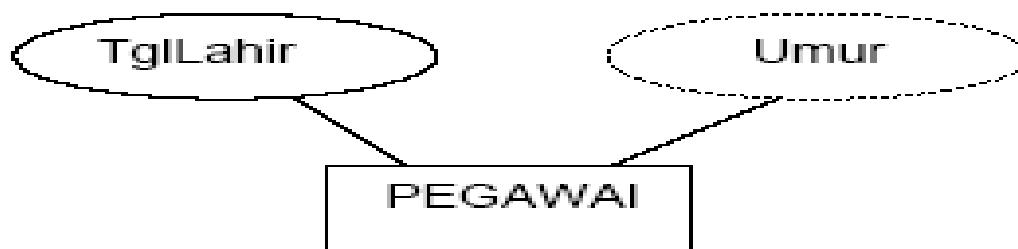
- Atribut Composite

Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.



- Atribut Derivatif

Suatu atribut yang dihasilkan dari atribut yang lain.



# RDBMS

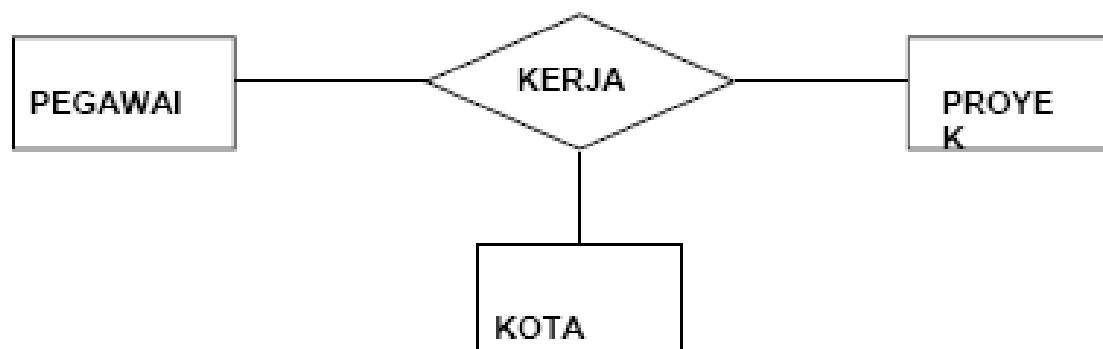
## a) Derajat Entity-Relationship

- Menjelaskan jumlah entity yang berpartisipasi dalam suatu relasi

Binary Degree (Derajat Dua)



Ternary Degree (Derajat Tiga)



# RDBMS

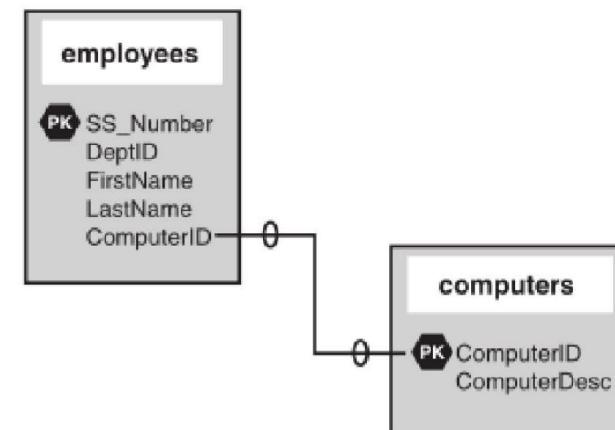
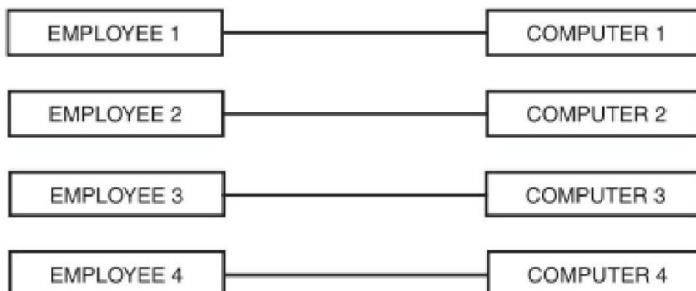
## a) ERD (Entity Relationship Diagram)

- ❑ Adalah **hubungan** antar **tabel** dalam database yang menciptakan keutuhan data.
- ❑ Diagram ER (Entity-Relationship) berisi kotak-kotak yang menyatakan entitas yang dihubungkan dengan garis-garis yang menunjukkan Relasi
- ❑ Ada 3 hubungan antar table: (Melani Julie C., 2004)
  1. One to one
  2. Many to one
  3. Many to many

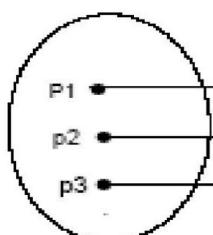
# RDBMS

## a) ERD - one to one

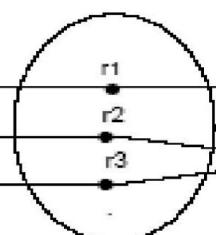
- ☐ A key appears **only** once in a **related** table. (Melani Julie C., 2004)



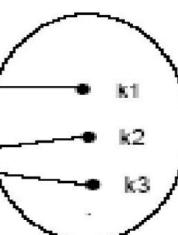
Pegawai



Milik



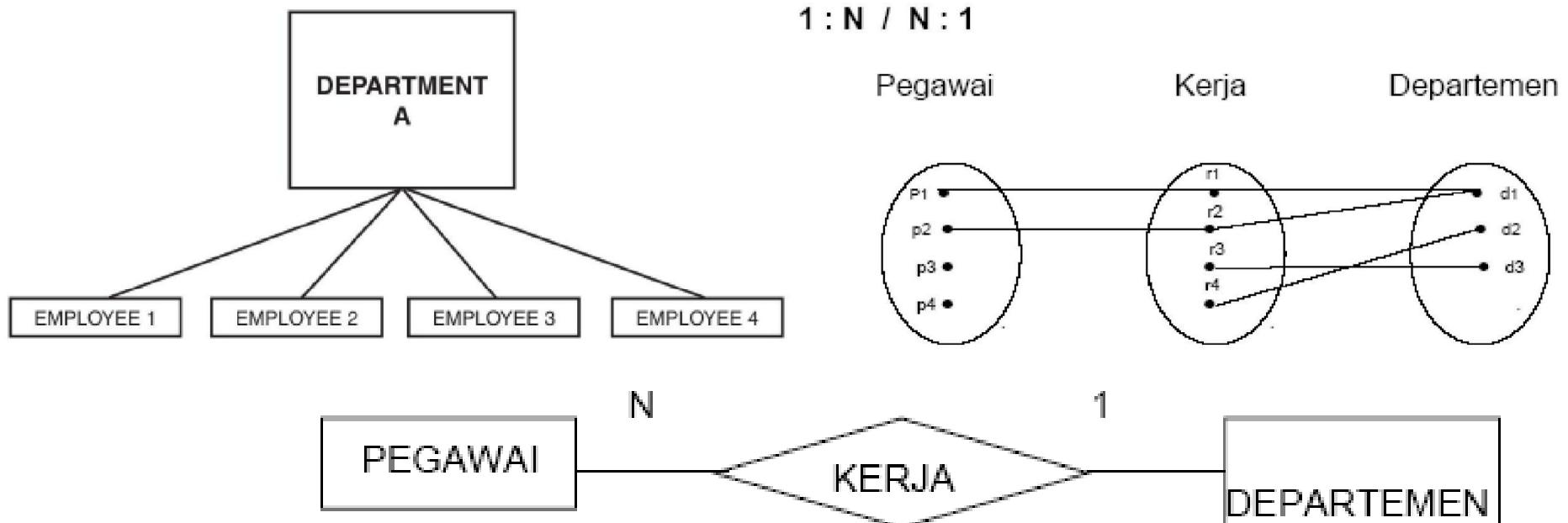
Kendaraan



# RDBMS

## a) ERD - one to many

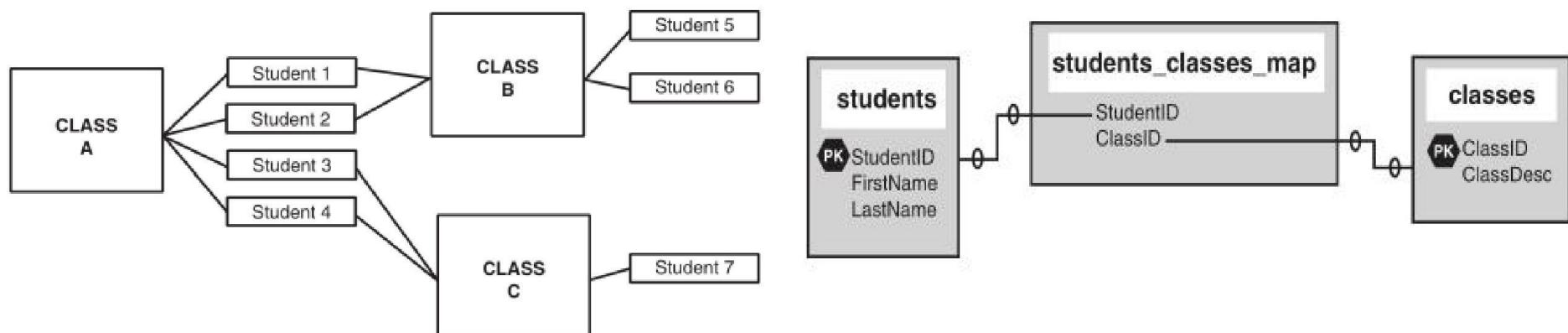
- ❑ Keys from one **table appear multiple times** in a related table. (Melani Julie C., 2004)
- ❑ Untuk setiap data di entitas pertama ada banyak data yang berhubungan di entitas kedua, tetapi untuk setiap data di entitas kedua ada satu dan hanya satu data di entitas pertama



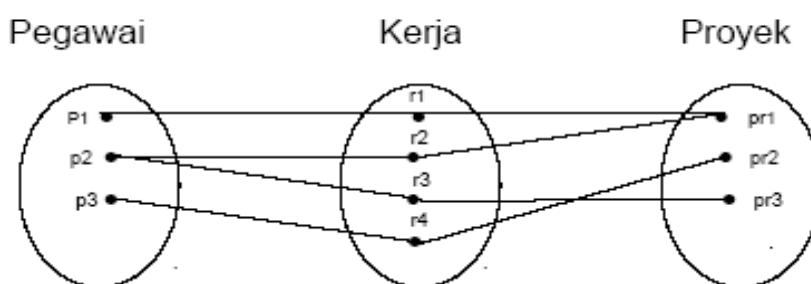
# RDBMS

## a) ERD - many to many

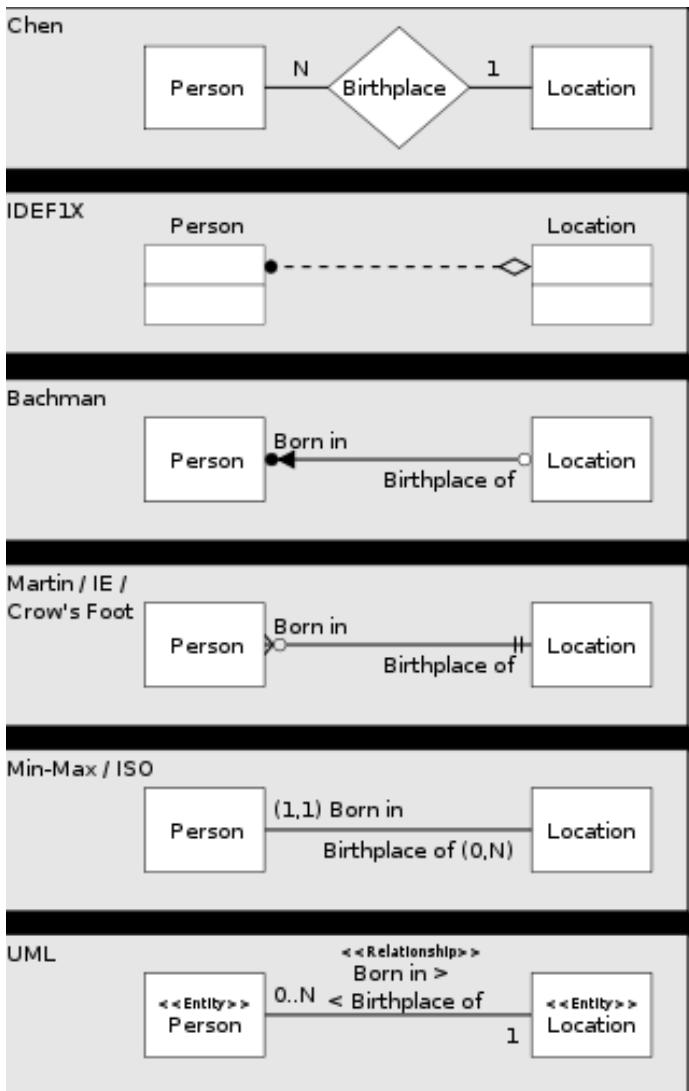
- Keys from one table appear multiple times in a related table. (Melani Julie C., 2004)
- Untuk setiap data di entitas pertama ada banyak data yang berhubungan di entitas kedua,begitu juga sebaliknya



M : N



# NOTASI ERD



Referensi definitif untuk entity relationship modelling secara umum, diulas pada tulisan Peter Chen (1976).

IDEFIX (Integration Definition for Information Modeling) bahasa pemodelan data untuk memodelkan data secara semantik, sebagai hasil dari program: Integrated Computer Aided Manufacturing (ICAM).

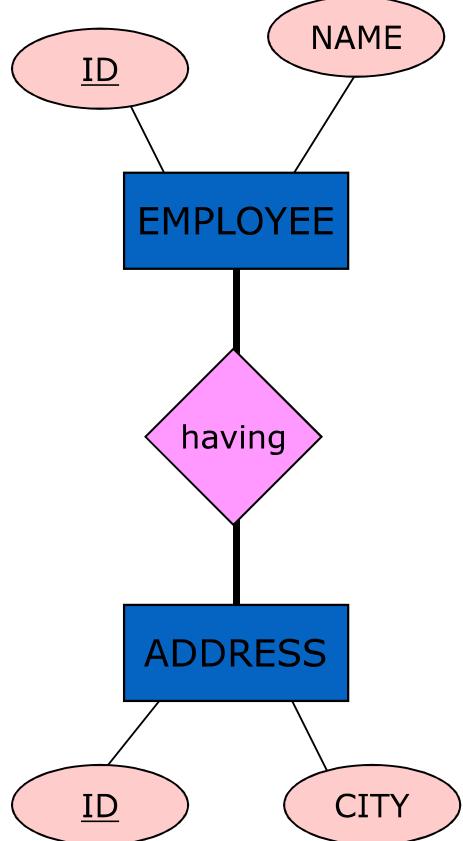
Notasi Bachman dari Charles Bachman.

Notasi Martin dari James Martin. Dinamakan juga notasi Crow's Foot, dan sangat populer.

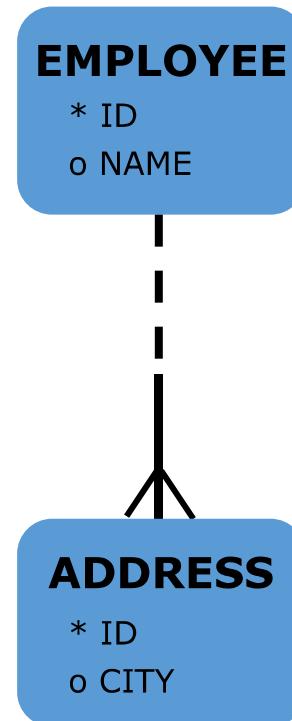
Notasi (min, max) dari Jean-Raymond Abrial pada 1974.

Notasi standard UML. Unified Modeling Language (UML) adalah bahasa yang digunakan untuk standarisasi pemodelan data pada software engineering.

# ERD DENGAN NOTASI CROW'S FOOT



Notasi  
Chen's



Notasi Crow's Foot  
menggunakan  
Oracle Designer

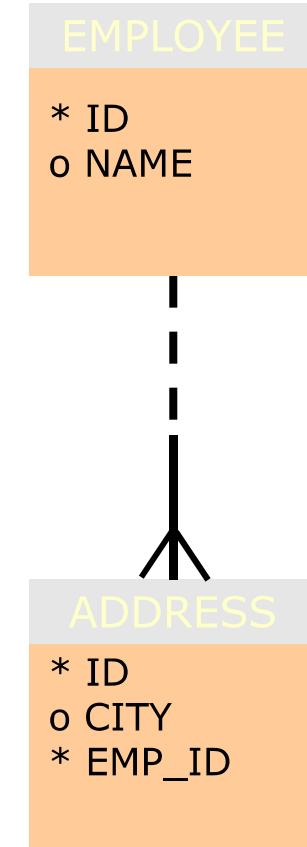
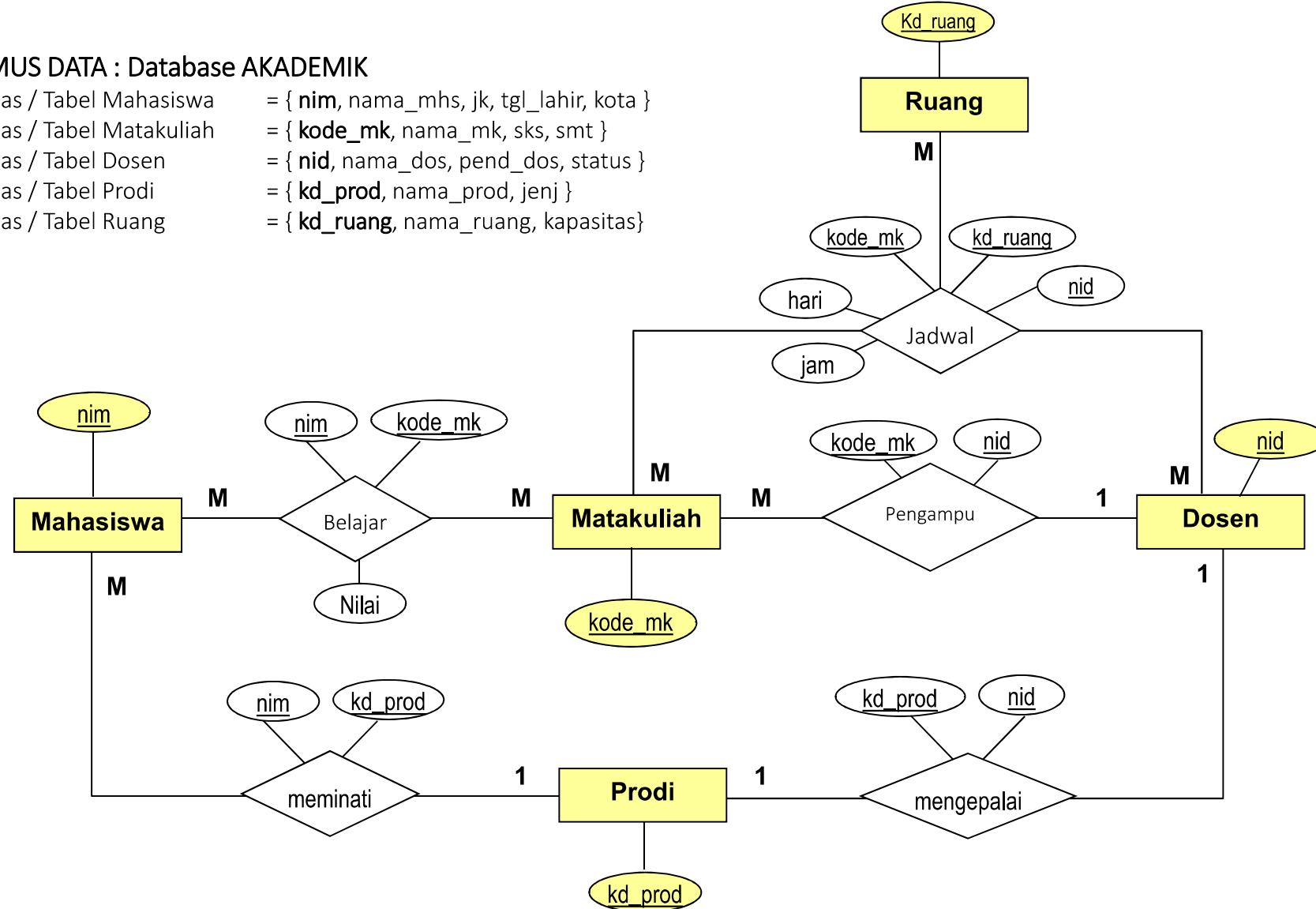


Table hasil

# Contoh ERD Database AKADEMIK

## KAMUS DATA : Database AKADEMIK

Entitas / Tabel Mahasiswa	= { nim, nama_mhs, jk, tgl_lahir, kota }
Entitas / Tabel Matakuliah	= { kode_mk, nama_mk, sks, smt }
Entitas / Tabel Dosen	= { nid, nama_dos, pend_dos, status }
Entitas / Tabel Prodi	= { kd_prod, nama_prod, jenj }
Entitas / Tabel Ruang	= { kd_ruang, nama_ruang, kapasitas}

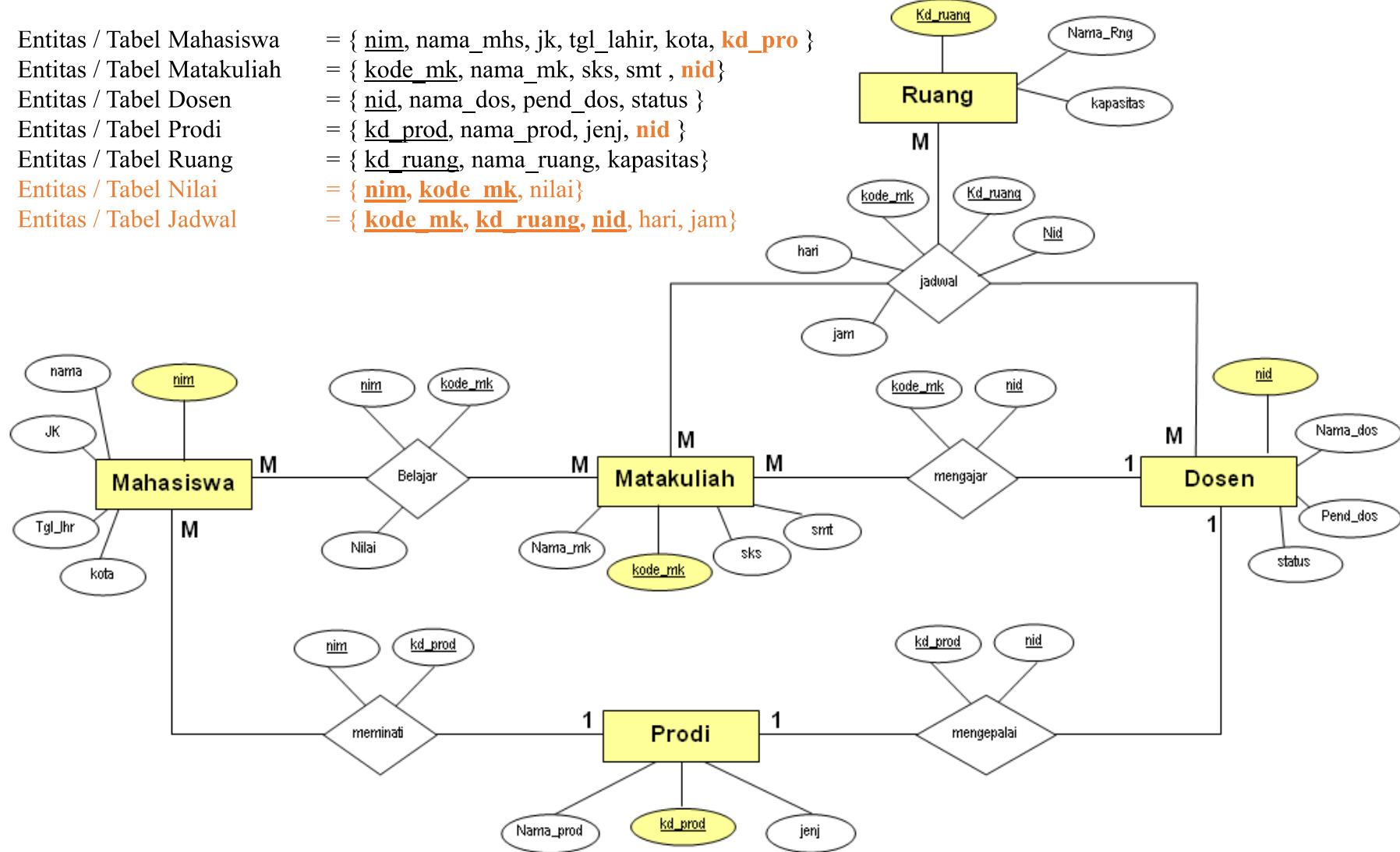


# ER-D lengkap dengan atribut (primary key & atribut deskriptif)

setelah adanya relasi antar tabel dengan derajat relasi maka struktur tabel berubah sebagai berikut :

Entitas / Tabel Mahasiswa  
 Entitas / Tabel Matakuliah  
 Entitas / Tabel Dosen  
 Entitas / Tabel Prodi  
 Entitas / Tabel Ruang  
 Entitas / Tabel Nilai  
 Entitas / Tabel Jadwal

= { nim, nama\_mhs, jk, tgl\_lahir, kota, **kd\_pro** }  
 = { kode\_mk, nama\_mk, sks, smt , **nid** }  
 = { nid, nama\_dos, pend\_dos, status }  
 = { kd\_prod, nama\_prod, jenj, **nid** }  
 = { kd\_ruang, nama\_ruang, kapasitas }  
 = { nim, kode\_mk, nilai }  
 = { kode\_mk, kd\_ruang, **nid**, hari, jam }



## KAMUS DATA : Database AKADEMIK (setelah ada relasi antar tabel)

- Entitas / Tabel Mahasiswa = { nim, nama\_mhs, jk, tgl\_lahir, kota, kd\_pro }  
 Entitas / Tabel Matakuliah = { kode\_mk, nama\_mk, skks, smt, nid }  
 Entitas / Tabel Dosen = { nid, nama\_dos, pend\_dos, status }  
 Entitas / Tabel Prodi = { kd\_prod, nama\_prod, jenj, nid }  
 Entitas / Tabel Ruang = { kd\_ruang, nama\_ruang, kapasitas }  
 Entitas / Tabel Nilai = { nim, kode\_mk, nilai }  
 Entitas / Tabel Jadwal = { kode\_mk, kd\_ruang, nid, hari, jam }

# Mapping dari ERD ke Skema Relasi

Tabel Mahasiswa

<u>nim</u>	<u>nama_mhs</u>	<u>jk</u>	<u>tgl_lahir</u>	<u>kota</u>	<u>kd_prod</u>
------------	-----------------	-----------	------------------	-------------	----------------

Tabel Ruang

<u>kd_ruang</u>	<u>nama_ruang</u>	<u>kapasitas</u>
-----------------	-------------------	------------------

Tabel Nilai

<u>nim</u>	<u>kode_mk</u>	<u>nilai</u>
------------	----------------	--------------

Tabel Jadwal

<u>kode_mk</u>	<u>kd_ruang</u>	<u>nid</u>	<u>hari</u>	<u>jam</u>
----------------	-----------------	------------	-------------	------------

Tabel Matakuliah

<u>kode_mk</u>	<u>nama_mk</u>	<u>skks</u>	<u>smt</u>	<u>nid</u>
----------------	----------------	-------------	------------	------------

Tabel Dosen

<u>nid</u>	<u>nama_dos</u>	<u>pend_dos</u>	<u>status</u>
------------	-----------------	-----------------	---------------

Tabel Program Studi

<u>kd_prod</u>	<u>nama_prod</u>	<u>jenj</u>	<u>nid</u>
----------------	------------------	-------------	------------

# Studi Kasus-Poliklinik

## Asumsi

- Setiap pasien yang akan memeriksakan kesehatan terdaftar dalam data pasien
- Setiap pasien memiliki catatan medik dari penyakit yang pernah diperiksakan, catatan medik juga menyimpan jenis penyakit dari pasien
- Setiap dokter memiliki catatan medik dari pasien-pasien yang telah diperiksa
- Poliklinik menangani transaksi pembelian obat dari pasien

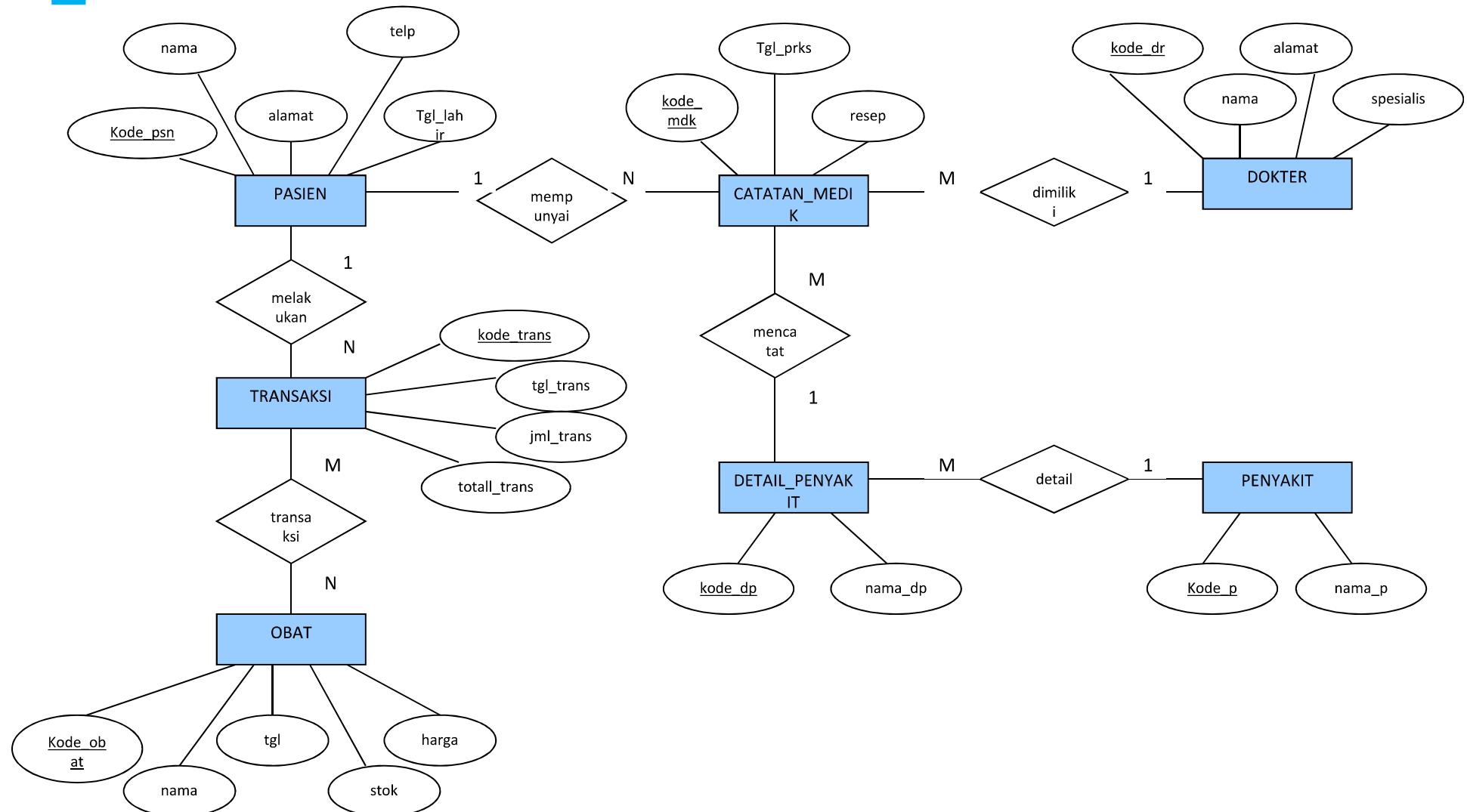
# Studi Kasus-Poliklinik

Entitas:

- Pasien
- Dokter
- Catatan\_medik
- Penyakit
- Detail\_penyakit
- Transaksi
- Obat



# ERD-Poliklinik



# RDBMS

## a) ERD - **Alur menentukan relasi tabel**

### □ Alur menentukan Relasi table:

1. Pahami system yang akan dibuat.
2. Berdasar point 1, ekstrak satu per satu tabelnya.
3. Relasikan tabel-tabel dengan terlebih dahulu menentukan hubungan antar tabelnya.

# RDBMS

## a) ERD - Studi Kasus 1

Sebuah sekolah dengan ketentuan:

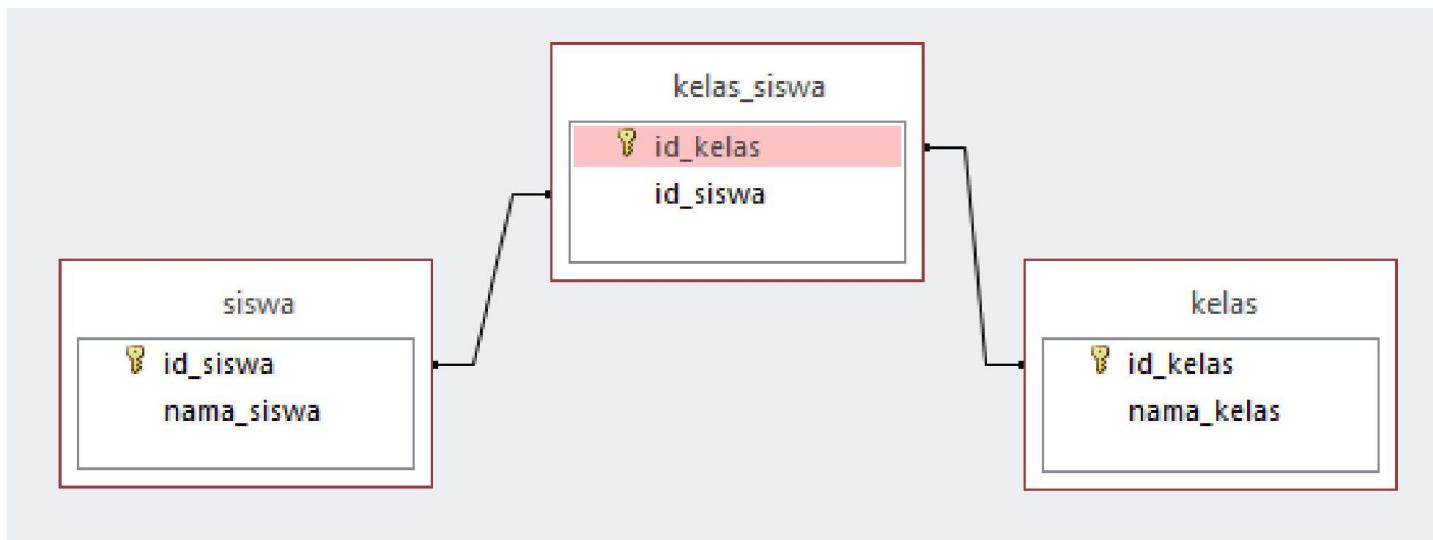
1. Setiap siswa akan masuk dalam sebuah kelas berjenjang.
2. Setiap kelas akan memiliki pelajaran yang telah ditetapkan.
3. Setiap pelajaran diajar oleh seorang guru.

# RDBMS

## a) ERD - Studi Kasus 1 (proses 1)

Sebuah sekolah dengan ketentuan:

1. Setiap siswa akan masuk dalam sebuah kelas berjenjang.

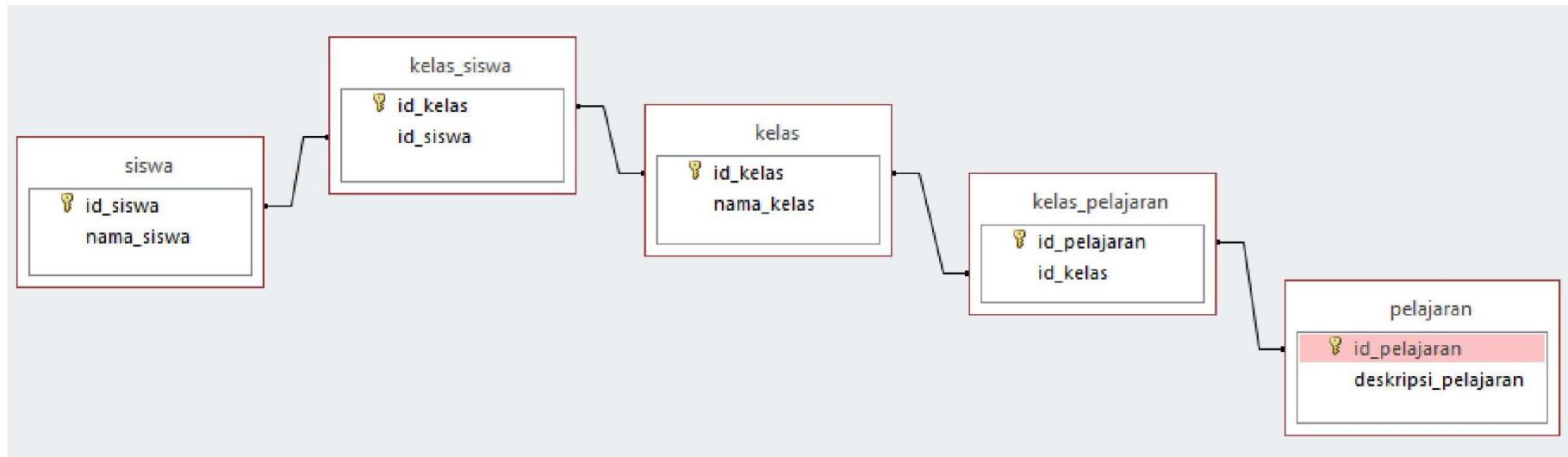


# RDBMS

## a) ERD - Studi Kasus 1 (proses 2)

Sebuah sekolah dengan ketentuan:

1. Setiap siswa akan masuk dalam sebuah kelas berjenjang.
2. Setiap kelas akan memiliki pelajaran yang telah ditetapkan.

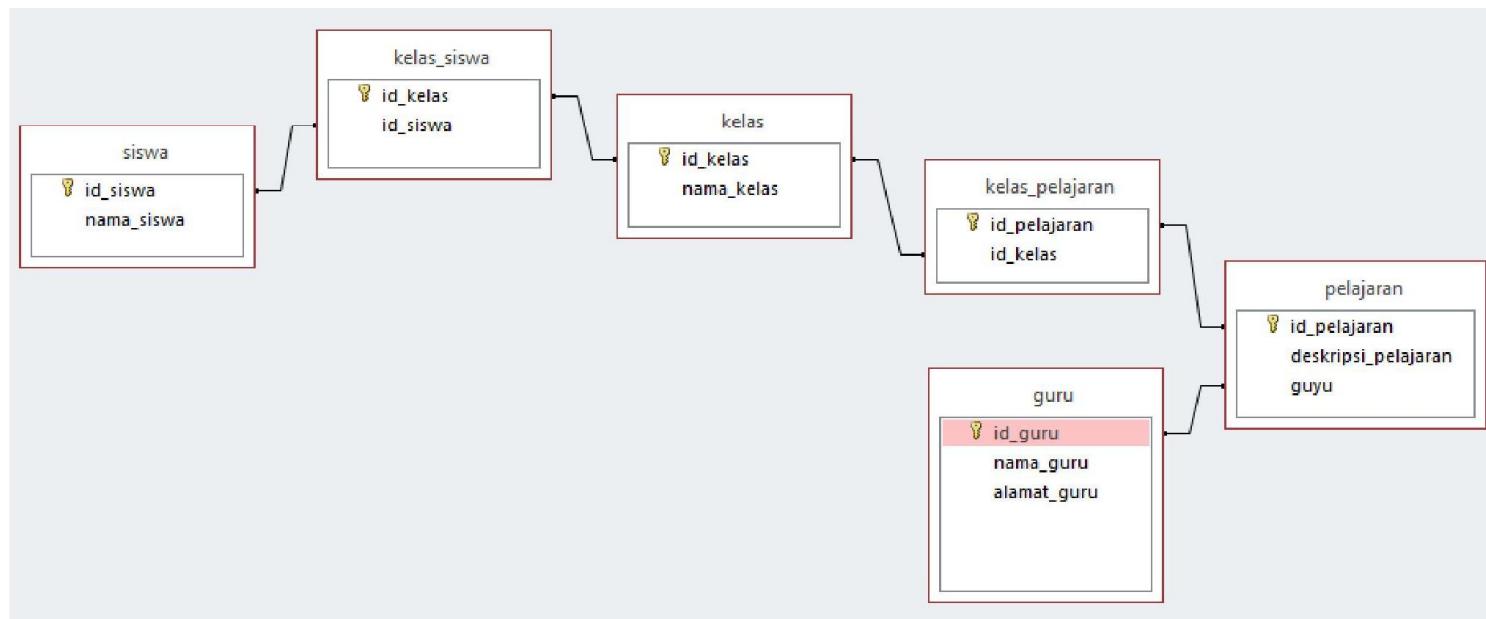


# RDBMS

## a) ERD - Studi Kasus 1 (proses 3)

Sebuah sekolah dengan ketentuan:

1. Setiap siswa akan masuk dalam sebuah kelas berjenjang.
2. Setiap kelas akan memiliki pelajaran yang telah ditetapkan.
3. Setiap pelajaran diajar oleh seorang guru.



# RDBMS

## a) ERD - Studi Kasus 2

Perpustakaan Fakultas Teknik akan membuat sebuah aplikasi dengan ketentuan:

- Anggota perpustakaan adalah mahasiswa, dosen, dan staff Fakultas Teknik.
- Lama waktu peminjaman buku untuk setiap anggota adalah 1 minggu.
- Jumlah maksimal peminjaman buku dalam satu waktu adalah 3 buah.
- Peminjaman dilayani oleh staff perpustakaan.
- Denda akan dikenakan jika peminjam melewati waktu pengembalian.
- Gambarkan relasi tabelnya!

# RDBMS

## a) ERD - Studi Kasus 2

Perpustakaan Fakultas Teknik akan membuat sebuah aplikasi dengan ketentuan:

- Anggota perpustakaan adalah mahasiswa, dosen, dan staff Fakultas Teknik. (Anggota)
- Lama waktu peminjaman buku untuk setiap anggota adalah 1 minggu. (tgl kembali, tgl pinjam)
- Jumlah maksimal peminjaman buku dalam satu waktu adalah 3 buah. (jml buku)
- Peminjaman dilayani oleh staff perpustakaan. (staff perpus)
- Denda akan dikenakan jika peminjam melewati waktu pengembalian. (denda)
- Gambarkan relasi tabelnya!**

# RDBMS

## a) ERD - Studi Kasus 2

### Entitas :

- Anggota
- Peminjaman
- Buku
- Staff Perpustakaan
- Denda

### Relasinya :

# RDBMS

## a) ERD - Studi Kasus 3

Seorang **dosen** akan **membuatkan** aplikasi bagi **seluruh dosen** di Fakultas Teknik UTM untuk **mengorganisir pengumpulan tugas** dari **kelas mahasiswanya**, dimana ketentuannya adalah sbb:

- Dibutuhkan **akses login** untuk memasuki aplikasi tersebut, dengan **menginputkan username dan password**.
- Tugas harus dikumpulkan tepat waktu.
- Pengumpulan tugas dapat dalam bentuk **pdf, teks, atau gambar**.
- Gambarkan relasi tabelnya!

# RDBMS

## a) ERD - Studi Kasus 3

### Entitas :

- Login (**nim,nip,user,pswd**)
- Mahasiswa (**nim,nama,jurusan,semester,user\_login**)
- Dosen (**nip, nama, jurusan,user\_login**)
- Tugas (kode tugas, jenis tugas, waktu pengumpulan)
- **Nilai** (**id\_tugas, id\_mahasiswa**)

### Relasinya :

# RDBMS

## a) ERD - Studi Kasus 4

- Sebuah supermarket memiliki pelayanan prima terhadap pelanggannya, salah satu wujudnya adalah dengan menyediakan kasir minimal 10.
- Setiap kasir bertugas melayani pelanggan yang akan melakukan pembelian barang-barang.
- Dalam satu shift juga terdapat 1 orang supervisor yang akan mengawasi 5 kasir dan membantu dalam pengangan khusus dari setiap transaksi yang terjadi.
- Pelayanan prima juga dilakukan dengan menerapkan 2 shift per harinya.
- Supermarket tersebut akan di SI-kan, maka gambarkan relasi tabelnya!

# RDBMS

## a) ERD - Studi Kasus 5

- Instansi dengan jumlah pengolahan data pegawai banyak membutuhkan bantuan SI.
- Pegawai yang didata mencakup PNS, Pensiuun (meninggal, pemberhentian dengan tidak hormat, pemberhentian dengan hormat), dan Non-PNS.
- Setiap pegawai akan memiliki riwayat terkait dengan pendidikan, keluarga, pangkat, dan lokasi kerjanya.
- Seluruh kegiatan dalam sistem akan ada log-nya, sehingga superadmin dapat mengetahui kegiatan di dalam Si tersebut.
- Orang yang akan menggunakan SI dengan username dan passwordnya adalah superadmin (pengatur sistem), admin (setiap skpd), dan pegawai.

# RDBMS

## a) ERD - Studi Kasus 6

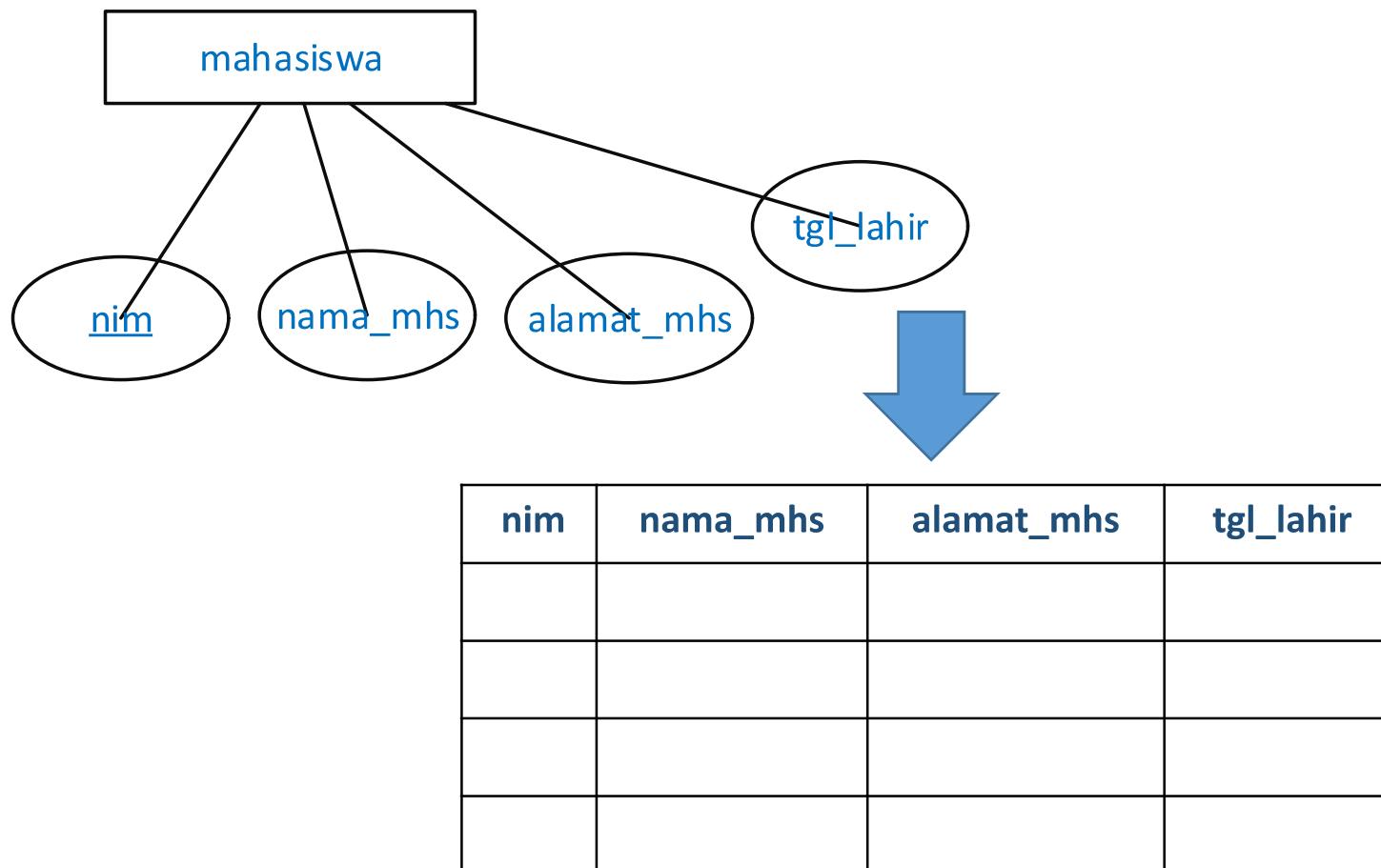
- Sekolah dengan keterbukaan dan ketertiban administrasi yang tinggi akan men-SI-kan pembayarannya.
- Setiap siswa yg membayar SPP akan dilayani oleh bendahara.
- Jumlah pembayaran siswa dalam tiap bulannya berkemungkinan berbeda antara kelas X, XI, dan XII.
- Pembayaran siswa boleh dilakukan secara cicil dalam tiap bulannya.
- Dalam pembayaran SPP terdapat beberapa rincian sub yang telah ditetapkan oleh sekolah.

# Penerapan Basis Data

## Transformasi himpunan entitas ke basis data fisik

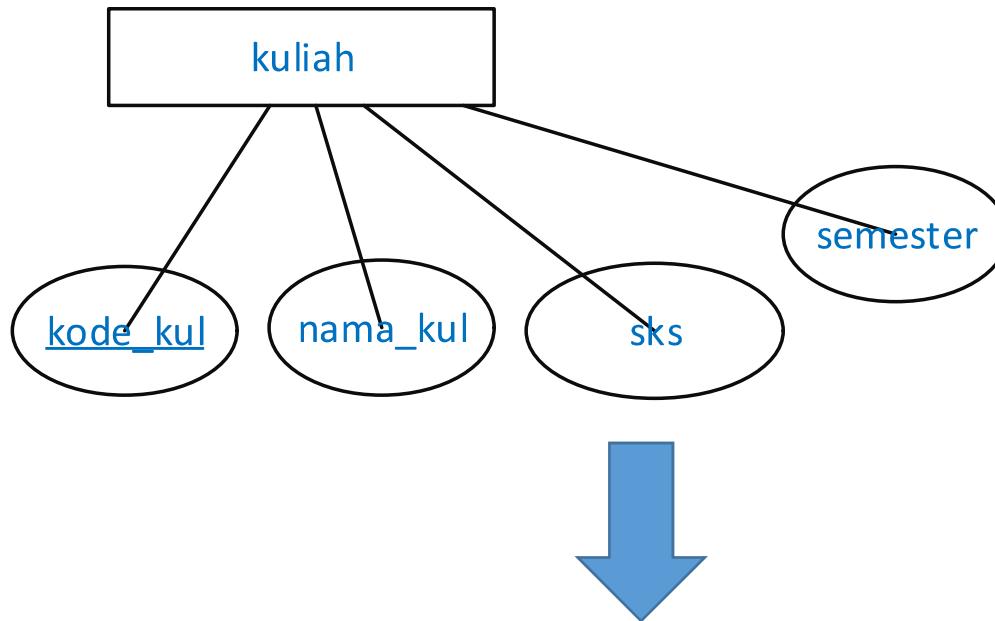
Aturan umum dalam pemetaan Model Data (Level konseptual dalam abstraksi data) :

1. Setiap himpunan entitas akan diimplementasikan sebagai sebuah tabel (file data)



# Penerapan Basis Data

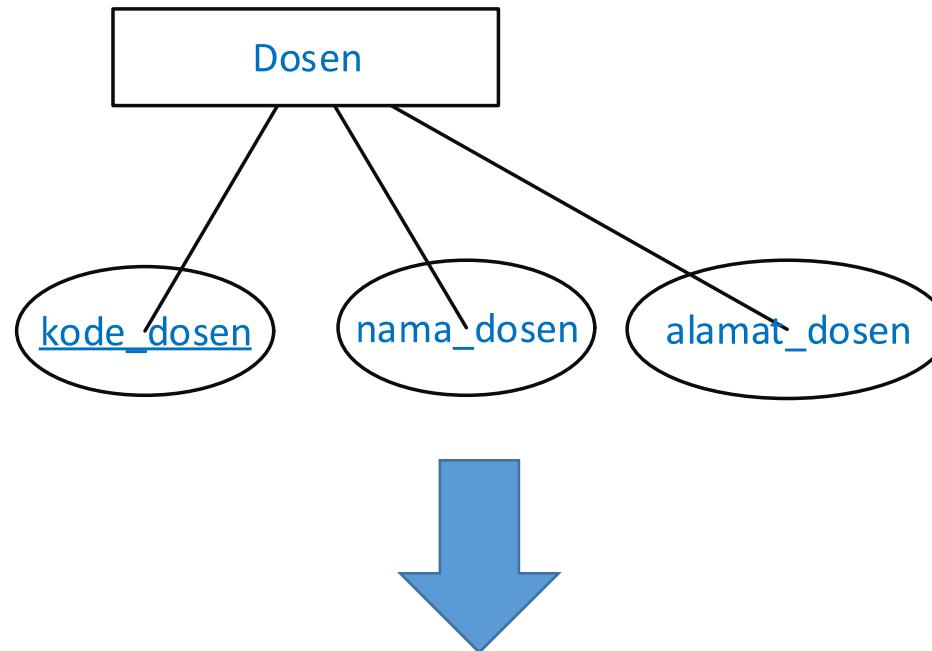
Transformasi himpunan entitas ke basis data fisik



Kode_kul	Nama_kul	skls	Semester

# Penerapan Basis Data

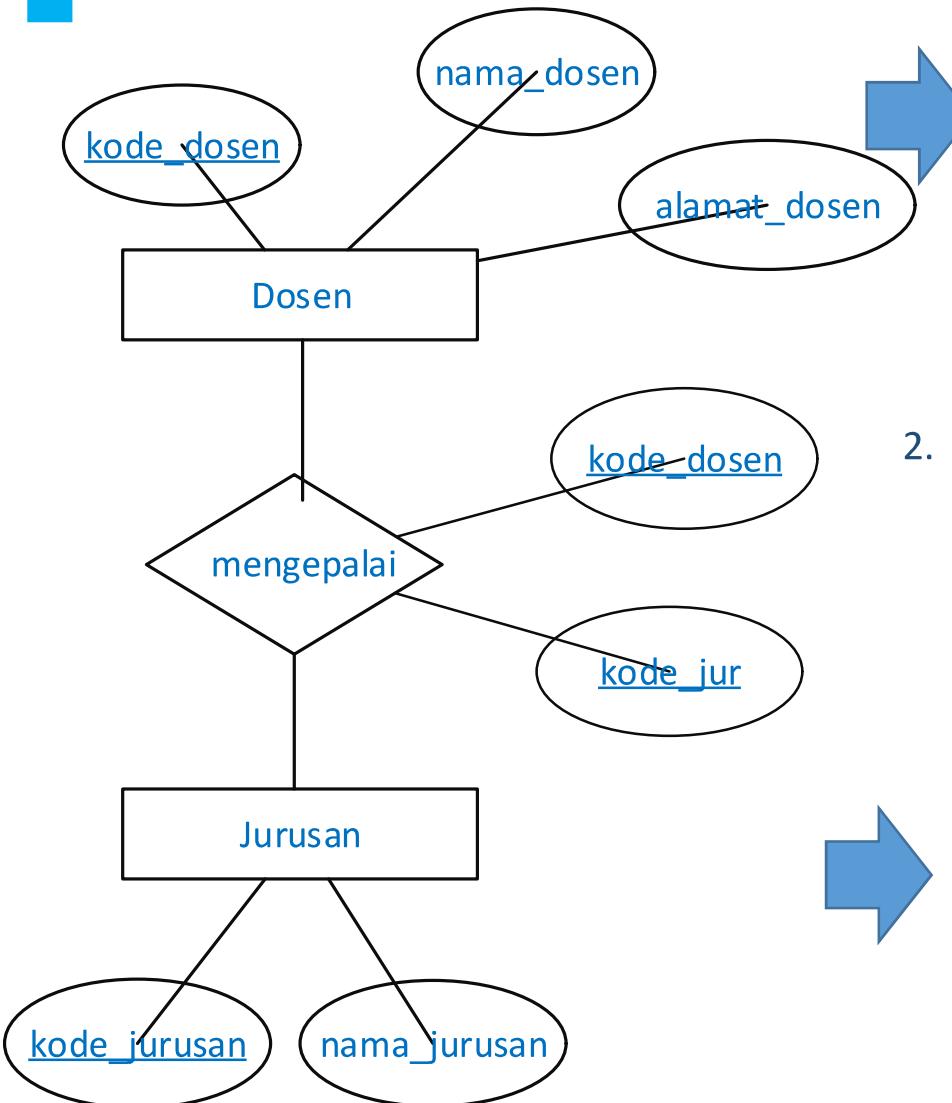
## Transformasi himpunan entitas ke basis data fisik



Kode_dosen	Nama_dosen	Alamat_dosen

# Penerapan Basis Data

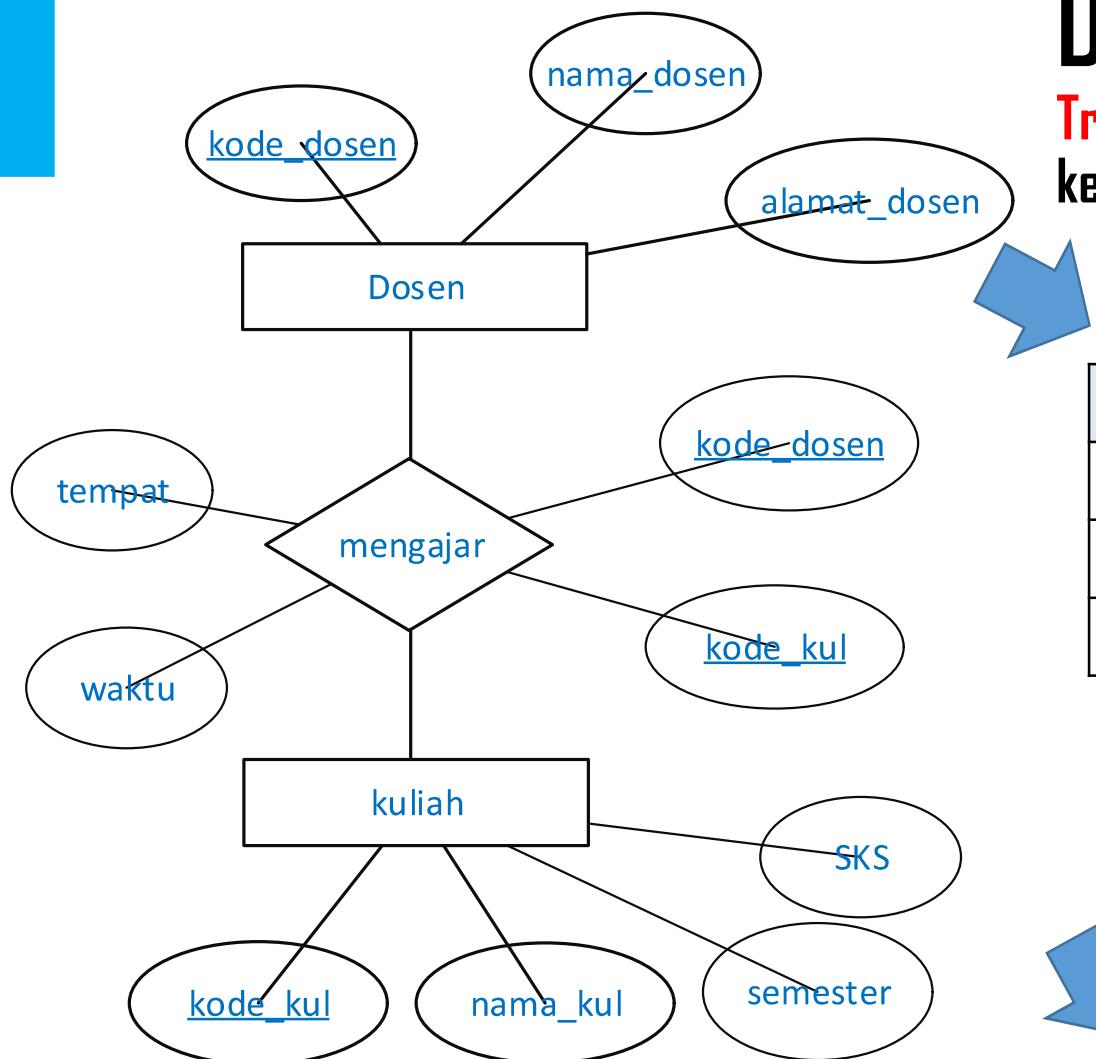
Transformasi relasi satu ke satu ke basis data fisik



2. Relasi dengan derajat relasi 1 – 1 yang menghubungkan dua buah himpunan entitas akan direpresentasikan dalam bentuk penambahan atribut relasi ke table yang mewakili salah satu dari kedua himpunan entitas.

# Penerapan Basis Data

Transformasi relasi satu kebanyak kebasis data fisik



Tabel Dosen

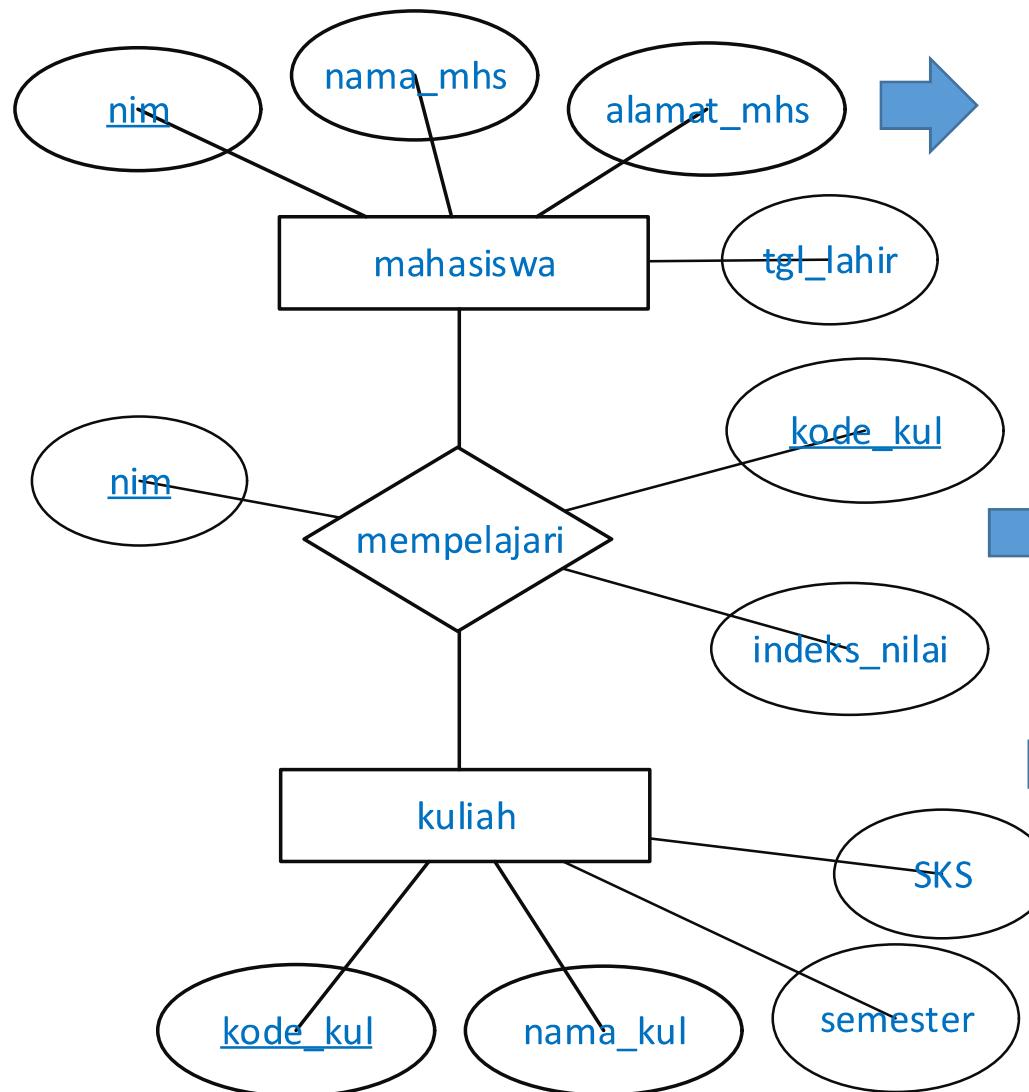
Kode_dosen	Nama_dosen	Alamat_dosen

Tabel Kuliah

Kode_kul	Nama_kul	skks	semester	Kode_dosen	tempat	waktu

# Penerapan Basis Data

Transformasi relasi banyak ke banyak ke basis data fisik



Tabel mahasiswa

nim	Nama_mhs	Alamat_mhs	Tgl_lahir

Tabel mempelajari/nilai

Tabel khusus yang mewakili himpunan relasi

nim	Kode_kul	Indeks_nilai

Tabel Kuliah

Kode_kul	Nama_kul	sks	semester

# 02. Database User Account Management

- 1) View
- 2) Grant
- 3) Revoke
- 4) Privelege
- 5) Adding User
- 6) Limiting User Resource

# SuperUser Vs Privileges

- User '**root**' dalam istilah keamanan komputer sering disebut sebagai '**superuser**'.
- **Superuser** user tertinggi dimana user ini dapat melihat, mengubah, bahkan menghapus seluruh database dan menjalankan perintah apapun yang terdapat dalam SQL (**CRUD**)
- **Privileges User** yang dibatasi hak aksesnya.
- Apakah user tersebut dapat membuat, mengubah dan menghapus sebuah tabel, atau user tersebut kita batasi hanya untuk melihat tabel saja (**SELECT**).

# Hak Akses

- **Hak akses** didalam Database adalah **hak** yang diberikan **kepada User** untuk dapat **mengakses data** / record tertentu.
- **Hak akses** ini jenisnya bermacam-macam bisa saja untuk memberikan **hak akses Tabel**, **hak akses Kolom** untuk dapat diakses oleh User tertentu.
- Setiap **user** dapat **dibatasi** untuk dapat **mengakses** baik itu sebuah **database** tertentu saja, **tabel** tertentu, atau bahkan hanya **kolom** tertentu.
- Kita dapat membuat user baru yang hanya bisa menjalankan perintah **SELECT** saja, dan user tersebut dibatasi untuk tidak dapat menjalankan query **DROP**.

# Level Hak Akses - GRANT

- Hak Akses Global (\*.\*)
- Hak Akses Level Database (nama\_database.\*)
- Hak Akses Level Tabel (nama\_database.nama\_tabel)
- Hak Akses Level Kolom (nama\_kolom)

# Hak Akses Global (\*.\*)

- ❑ Hak akses yang dapat mengakses seluruh bagian didalam suatu database
- ❑ Misalkan Tabel, Kolom, dan Data.
- ❑ Hak akses jenis ini dapat melakukan apapun CRUD didalam Database.
- ❑ Penulisan querynya biasanya (\*.\*).
- ❑ Untuk memberi akses terhadap user dilakukan di User Root.
- ❑ **Sql :**  
**GRANT hak\_akses ON \*.\* TO "nama\_user"@"lokasi\_user";**  
**GRANT SELECT ON \*.\* TO 'JUN1'@'localhost', 'EVE1'@'localhost';**

# Hak Akses Database - nama\_database.\*

- ❑ Hanya dapat mengakses Database tertentu saja serta dapat mengakses seluruh Tabel dan Kolom didalam Database tersebut.
- ❑ Sql :

```
GRANT hak_akses ON nama_database.* TO "nama_user"@"lokasi_user";
```

```
GRANT SELECT ON tennis.* TO 'JUN1'@'localhost';
```

# Hak Akses Tabel - nama\_database.nama\_tabel

- User memiliki hak akses pada sebuah tabel beserta Kolomnya yang berada pada sebuah database.
- Hak akses yang dimiliki user hanya terbatas pada level sebuah tabel saja.
- Sql :

```
GRANT hak_akses ON nama_database . nama_tabel TO  
"nama_user"@"lokasi_user";
```

```
GRANT SELECT ON tennis.players TO 'JUN1'@'localhost';
```

# Hak Akses Level Kolom - nama\_kolom

- Hak akses ini adalah hak akses paling kecil yang dapat diberikan kepada sebuah user.
- Dengan hak akses level kolom, user hanya memiliki hak akses untuk beberapa kolom pada sebuah tabel.
- Level paling akhir ini kita membatasi hak akses user hanya untuk kolom tertentu saja.
- Penulisan kolom yang diperbolehkan diletakkan di dalam tanda kurung.
- Sql :

```
GRANT hak_akses (nama_kolom) ON nama_database.nama_tabel  
TO "nama_user"@"lokasi_user";
```

```
GRANT SELECT (nama,umur) ON tennis.players TO 'JUN1'@'localhost';
```

# Bentuk otorisasi dalam basis data

- **Read Authorization** – pengguna diperbolehkan membaca data, tetapi tidak dapat memodifikasi.
- **Insert Authorization** – pengguna diperbolehkan menambah data baru, tetapi tidak dapat memodifikasi data yang sudah ada.
- **Update Authorization** – pengguna diperbolehkan memodifikasi data, tetapi tidak dapat menghapus data.
- **Delete Authorization** – pengguna diperbolehkan menghapus data.

# Bentuk otorisasi untuk modifikasi skema basis data

- **Index Authorization** = pengguna diperbolehkan membuat dan menghapus **index** data.
- **Authorization** = pengguna diperbolehkan **membuat relasi-relasi baru**.
- **Alteration Authorization** = pengguna diperbolehkan **menambah/menghapus atribut suatu relasi**.
- **Drop Authorization** = pengguna diperbolehkan **menghapus relasi** yang sudah ada.

# Otorisasi dan View

- **View** adalah **objek basis data** yang berisi perintah **query** ke basis data
- Setiap kali sebuah **view diaktifkan**, pemakai akan selalu **melihat hasil querynya**.
- Berbeda dengan tabel, **data** yang ditampilkan **didalam view tidak bisa di ubah**.
- **View** menyediakan mekanisme **pengamanan yang fleksibel** namun **kuat** dengan cara **menyembunyikan sebagian basis data** dari user lain
- **User** dapat **diberikan otorisasi pada View**, tanpa harus diberikan otorisasi terhadap relasi yang digunakan di dalam **definisi view**.
- **View** dapat **meningkatkan keamanan data** dengan **mengizinkan user** untuk **hanya dapat mengakses data** sesuai dengan **pekerjaannya masing-masing**.
- **Kombinasi** antara **level keamanan** ditingkat **relasional** dengan **level keamanan** **dingkat view** dapat digunakan untuk **membatasi hak akses user**, sehingga mereka **hanya mengakses data** sesuai dengan **kebutuhan** saja.

# VIEWS

```
CREATE VIEW view_name (column_name) AS  
[SELECT BLOCK] [WITH CHECK OPTION]
```

- Merupakan '*derivedtables*' ⇒ harus didefinisikan dalam bentuk query pada tabel atau view yang lain
- Merupakan '*virtual tables*' ⇒ tidak akan dievaluasi, kecuali jika mereka digunakan dalam query lain
- reusable
- Dapat digunakan untuk membuat query yang sulit (atau bahkan tidak mungkin) untuk dilakukan dalam suatu kondisi tertentu perintah INSERT,
- UPDATE, atau DELETE dapat dilakukan terhadap data yang ada di dalam tabel basis melalui view tabel basis tersebut

# VIEWS

**CREATE VIEW view\_name (column\_name) AS  
[SELECT BLOCK] [WITH CHECK OPTION]**

- ❑ Tabel View bisa berasal dari tabel lain, atau gabungan dari beberapa tabel.
- ❑ Tujuan :
  - ❑ kenyamanan (mempermudah penulisan query),
  - ❑ Keamanan (menyembunyikan beberapa kolom yang bersifat rahasia),
  - ❑ beberapa kasus bisa digunakan untuk mempercepat proses menampilkan data (terutama jika kita akan menjalankan query tersebut secara berulang)

# Create VIEWS

- Contoh 1:** Buatlah **view** untuk membuat daftar seluruh nama kota yang ada dalam tabel **PLAYERS**!
- CREATE VIEW TOWNS (TOWN)  
AS SELECT DISTINCT TOWN  
FROM PLAYERS**
- Sintaks Alternatif:
- CREATE VIEW TOWNS AS  
SELECT DISTINCT TOWN FROM  
PLAYERS**
- Query SELECT:
- SELECT \* FROM TOWNS**

1	CREATE VIEW TOWNS AS SELECT DISTINCT TOWN FROM PLAYERS
/	players (1x6)
TOWN	
Stratford	
Inglewood	
Eltham	
Midhurst	
Douglas	
Plymouth	

# Create VIEWS (contd 2)

- Contoh 2: Buatlah view untuk membuat daftar nomor pemain dan nomor liga dari seluruh pemain yang memiliki nomor liga!
- **CREATE VIEW CPLAYERS AS SELECT PLAYERN0, LEAGUENO FROM PLAYERS WHERE LEAGUENO IS NOT NULL**
- Query SELECT:
- **SELECT \* FROM CPLAYERS**
- Contoh 3: Dapatkan seluruh informasi mengenai pemain yang memiliki nomor liga yang nomor pemainnya adalah antara 6 dan 44!
- **SELECT \* FROM CPLAYERS WHERE PLAYERN0 BETWEEN 6 AND 44**

```
1 CREATE VIEW CPLAYERS AS SELECT PLAYERN0, LEAGUENO FROM PLAYERS WHERE LEAGUENO IS NOT NULL
2 SELECT * FROM CPLAYERS
3
```

CPLAYERS (2x10)	
PLAYERN0	LEAGUENO
2	2411
6	8467
8	2983
27	2513
44	1124
57	6409
83	1608
100	6524
104	7060
112	1319

```
1 SELECT * FROM CPLAYERS WHERE PLAYERN0 BETWEEN 6 AND 44
```

CPLAYERS (2x4)	
PLAYERN0	LEAGUENO
6	8467
8	2983
27	2513
44	1124

# Create VIEWS (contd 3)

- Contoh 4: Buatlah **view** untuk membuat **daftar seluruh pemain** yang memiliki **nomor liga** dengan **nomor pemain** antara **6 dan 27!**
- **CREATE VIEW SEVERAL AS SELECT \* FROM CPLAYERS WHERE PLAYERNO BETWEEN 6 AND 27**

1 `SELECT * FROM CPLAYERS`

PLAYERNO	LEAGUENO
2	2411
6	8467
8	2983
27	2513
44	1124
57	6409
83	1608
100	6524
104	7060
112	1319

1 `SELECT * FROM SEVERAL`

PLAYERNO	LEAGUENO
6	8467
8	2983
27	2513

# Create VIEWS (contd 4)

- Contoh 5: Tentukan **(nilai) rata-rata** dari **keseluruhan penalti** yang pernah dilakukan oleh **pemain!**
- Solusi berikut **tidak mungkin** dapat dilakukan:
- **SELECT PLAYERNO, AVG(SUM(AMOUNT)) FROM PENALTIES GROUP BY PLAYERNO;**
- Solusiyang dapat dilakukan dengan menggunakan VIEW:
- **CREATE VIEW SUM\_PENALTIES (PLAYERNO, SUM\_AMOUNT) AS SELECT PLAYERNO, SUM(AMOUNT) FROM PENALTIES GROUP BY PLAYERNO;**
- **SELECT AVG(SUM\_AMOUNT) FROM SUM\_PENALTIES;**

```
1 CREATE VIEW SUM_PENALTIES (PLAYERNO, SUM_AMOUNT) AS  
2 SELECT PLAYERNO, SUM(AMOUNT) FROM PENALTIES GROUP BY PLAYERNO;
```

# Create VIEWS (contd 5)

- Sintaks Alternatif untuk membuat VIEW:
- **CREATE VIEW**  
**SUM\_PENALTIES AS**  
**SELECT PLAYERNO,**  
**SUM(AMOUNT) AS**  
**SUM\_AMOUNT FROM**  
**PENALTIES GROUP BY**  
**PLAYERNO;**

```
1 SELECT AVG(SUM_AMOUNT)
2 FROM SUM_PENALTIES;
```

Hasil #1 (1x1)

AVG(SUM_AMOUNT)
96,000000

# Create VIEWS (contd 5)

- Contoh 6: Buatlah **view** untuk **membuat daftar** urutan digit dari 0 sampai 9!

- **CREATE VIEW DIGITS AS**

```
SELECT 0 DIGIT UNION
```

```
SELECT 1 UNION
```

```
SELECT 2 UNION
```

```
SELECT 3 UNION
```

```
SELECT 4 UNION
```

```
SELECT 5 UNION
```

```
SELECT 6 UNION
```

```
SELECT 7 UNION
```

```
SELECT 8 UNION
```

```
SELECT 9
```

- **SELECT \* FROM DIGITS**

```
1 | SELECT * FROM DIGITS
```

-hasil #1 (1×10)

DIGIT
0
1
2
3
4
5
6
7
8
9

# Create VIEWS -- WITH CHECK OPTION

- CREATE VIEW VETERANS AS SELECT \* FROM PLAYERS WHERE BIRTH\_DATE < '1960-01-01';**
- SELECT \* FROM VETERANS;**

```
1 SELECT * FROM VETERANS;
```

- DELAY
- DELET
- DESC
- DETER
- DIREC

VETERANS (12x3)												
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO	
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411	
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)	
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608	

# Create VIEWS (contd 6)

- Contoh 7 : Misalkan kita ingin menampilkan nama pemain yang berjenis kelamin female.

```
SELECT PLAYERO NO, NAME, INITIALS , sex, TOWN FROM  
players WHERE sex='F'
```

```
1 SELECT PLAYERO NO, NAME, INITIALS , sex, TOWN FROM players WHERE sex='F'  
2 |
```

players (5x5)				
PLAYERO NO	NAME	INITIALS	sex	TOWN
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

Bagaimana jika query tersebut akan dijalankan setiap beberapa detik (diakses dari website yang sibuk)???

**Solusinya? view**

# Create VIEWS (contd 7)

- Contoh 8 : Solusinya dengan membuat View untuk menampilkan nama pemain yang berjenis kelamin female. Sehingga beban server berkurang dan VIEW dapat menyembunyikan beberapa kolom dari tabel players.

```
CREATE VIEW sex_female AS SELECT  
PLAYERNO, NAME, INITIALS, sex,  
TOWN FROM players WHERE sex='F'
```

```
SELECT * FROM sex_female
```

```
SELECT name FROM sex_female  
WHERE playerno='8'
```

SELECT \* from sex\_female

sex_female (5x5)				
PLAYERNO	NAME	INITIALS	sex	TOWN
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

```
1 SELECT name FROM sex_female WHERE playerno='8'
```

sex\_female (1x1)

NAME
Newcastle

# Create VIEWS (contd 8)

- Bagaimana jika tabel utama di update?

```
1 SELECT * from players
```

players (12x14)												
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO	
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411	
6	Parmenter	R	1964-06-25	M	1.977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467	
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)	
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983	
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513	
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)	
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)	
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124	
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409	
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812UP	Stratford	070-353548	1608	
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)	
100	Parmenter	P	1963-02-28	M	1.979	Haseltine Lane	80	6494SG	Stratford	070-494593	6524	
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060	
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319	

# Create VIEWS (contd 9)

**INSERT INTO PLAYERS VALUES**

```
(3, 'Dian', 'Di', '1999-05-20', 'F', 2000, 'Surabaya', '11', '60208', 'Surabaya', '0856-4868-8777', '12'),
(4, 'Diana', 'Da', '1999-06-21', 'F', 2000,
'Bojonegoro', '12', '62115', 'Bojonegoro',
'0800-0000-1111', '13'),
(5, 'Dinda', 'Dn', '1999-07-22', 'F', 2000,
'Semarang', '13', '63115',
'Semarang', '0811-1111-0000', '14');
```

1 SELECT * FROM players												
players (12x17)												
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO	
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	5579NH	Stratford	070-237893	2411	
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12	
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13	
5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Semarang	0811-1111-	14	
6	Parmenter	R	1964-06-25	M	1.977	Haseline Lane	80	1234KX	Stratford	070-476537	8467	
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9759VB	Stratford	070-347680	(NULL)	
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WQ	Inglewood	070-458458	2883	
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457OK	Eltham	079-234857	2513	
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294KX	Midhurst	010-659599	(NULL)	
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9529CD	Stratford	070-393435	(NULL)	
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124	
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409	
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	16A	1812JP	Stratford	070-353548	1608	
95	Miller	P	1963-05-14	M	1.972	High Street	35A	5746OP	Douglas	070-867564	(NULL)	
100	Parmenter	P	1963-02-28	M	1.979	Haseline Lane	80	6494SG	Stratford	070-494593	6524	
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060	
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319	

1 SELECT * FROM sex_female				
sex_female (5x8)				
PLAYERNO	NAME	INITIALS	sex	TOWN
3	Dian	Di	F	Surabaya
4	Diana	Da	F	Bojonegoro
5	Dinda	Dn	F	Semarang
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth

Terlihat bahwa VIEW juga otomatis diupdate.

# Create VIEWS (contd 10)

- Apakah kita bisa menambahkan data ke dalam VIEW? Nama Viewnya : sex\_female

```
INSERT INTO sex_female VALUES  
(200, 'Sisi',  
'Ss', 'F', 'Madura');
```

Ternyata ketika kita update pada VIEW. Pada tabel asal juga mengalami Update meskipun akan terdapat nilai **NULL** di dalam tabel asli

1 SELECT * FROM sex_female				
sex_female (5x9)				
PLAYERNO	NAME	INITIALS	sex	TOWN
3	Dian	Di	F	Surabaya
4	Diana	Da	F	Bojonegoro
5	Dinda	Dn	F	Semarang
8	Newcastle	B	F	Inglewood
27	Collins	DD	F	Eltham
28	Collins	C	F	Midhurst
104	Moorman	D	F	Eltham
112	Bailey	IP	F	Plymouth
200	Sisi	Ss	F	Madura

1 SELECT * FROM players											
players (12x18)											
PLAYERNO	NAME	INITIALS	BIRTH_DATE	SEX	JOINED	STREET	HOUSENO	POSTCODE	TOWN	PHONENO	LEAGUENO
2	Everett	R	1948-09-01	M	1.975	Stoney Road	43	3575NH	Stratford	070-237893	2411
3	Dian	Di	1999-05-20	F	2.000	Surabaya	11	60208	Surabaya	0856-4868-	12
4	Diana	Da	1999-06-21	F	2.000	Bojonegoro	12	62115	Bojonegoro	0800-0000-	13
5	Dinda	Dn	1999-07-22	F	2.000	Semarang	13	63115	Semarang	0811-1111-	14
6	Parmenter	R	1964-06-25	M	1.977	Haseline Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963-05-11	M	1.981	Edgecombe Way	39	9758VB	Stratford	070-347689	(NULL)
8	Newcastle	B	1962-07-08	F	1.980	Station Road	4	6584WO	Inglewood	070-458458	2983
27	Collins	DD	1964-12-28	F	1.983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963-06-22	F	1.983	Old Main Road	10	1294QK	Midhurst	010-659599	(NULL)
39	Bishop	D	1956-10-29	M	1.980	Eaton Square	78	9629CD	Stratford	070-393435	(NULL)
44	Baker	E	1963-01-09	M	1.980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971-08-17	M	1.985	Edgecombe Way	15	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956-11-11	M	1.982	Magdalene Road	15A	1812UP	Stratford	070-353548	1608
95	Miller	P	1963-05-14	M	1.972	High Street	33A	5746OP	Douglas	070-867564	(NULL)
100	Parmenter	P	1963-02-28	M	1.979	Haseline Lane	80	6494SG	Stratford	070-494593	6524
104	Moorman	D	1970-05-10	F	1.984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963-10-01	F	1.984	Vixen Road	8	6392LK	Plymouth	010-548745	1319
200	Sisi	Ss	(NULL)	F	0	(NULL)	(NULL)	(NULL)	Madura	(NULL)	(NULL)

# USERS AUTHORISATION

- Untuk proteksi data: USER ,PASSWORD, HAK AKSES
- Hak Akses meliputi:
  - Hak akses terhadap suatu **Kolom** tertentu dari suatu tabel
  - Hak akses terhadap suatu **Tabel**
  - Hak akses terhadap tabel-tabel yang ada pada suatu **Basisdata** tertentu
- Hak akses suatu **User** terhadap seluruh basisdata yang ada di dalam server

# Menambah, Melihat & Menghapus USERS

- **CREATE USER 'user\_name'@'host\_name' IDENTIFIED BY password\_user**
- Contoh 1: Buatlah dua user baru, yaitu **EVE** dengan password **EVE\_PASS**, dan **JUN** dengan password **JUN\_PASS**!
- **CREATE USER**  
`'EVE'@'localhost' IDENTIFIED BY 'EVE_PASS',  
'JUN'@'localhost' IDENTIFIED BY 'JUN_PASS'`

Melihat user :

```
SELECT user,host FROM mysql.user;
```

user (2x7)	
user	host
root	127.0.0.1
root	::1
	localhost
EVE	localhost
<b>JUN</b>	localhost
pma	localhost
root	localhost

# Menambah, Melihat & Menghapus USERS

- DROP USER 'user\_name'@'host\_name'
- **Contoh 2:** Hapuslah user JUN!

```
DROP USER 'JUN'@'localhost'
```

1 SELECT user,host FROM mysql.USER;	
user	host
root	127.0.0.1
root	::1
	localhost
EVE	localhost
pma	localhost
root	localhost

```
DROP USER  
'JUN'@'localhost', 'EVE'@'localho  
st'
```

# Mengubah Nama & Password USERS

- **RENAME USER** 'user\_name'@'host\_name' TO 'user\_name'@'host\_name'
- Contoh 3: **Ubahlah** nama user **EVE** dan **JUN** menjadi **EVE1** dan **JUN1**

**RENAME USER**

```
'EVE'@'localhost' TO 'EVE1'@'localhost',  
'JUN'@'localhost' TO 'JUN1'@'localhost'
```

- **SET PASSWORD FOR** 'user\_name'@'host\_name' = **PASSWORD** ('new\_password')
- Contoh 4: **Ubahlah** password user **JUN1** menjadi **JUN1\_PASS!**

```
SET PASSWORD FOR 'JUN1'@'localhost' = PASSWORD ('JUN1_PASS')
```

# Hak Akses : Tabel & Kolom

- **Select** : User berhak untuk mengakses tabel tertentu dengan menggunakan pernyataan SELECT
- **INSERT** : user berhak untuk menambah baris pada tabel tertentu dengan menggunakan pernyataan INSERT
- **DELETE** : user berhak untuk menghapus baris dari tabel tertentu dengan menggunakan pernyataan DELETE
- **UPDATE** : user berhak untuk mengubah nilai dari suatu tabel tertentu dengan menggunakan pernyataan UPDATE
- **REFERENCES** : user berhak untuk membuat Foreign Key ( FK ) yang mengacu pada suatu tabel
- **CREATE** : user berhak untuk membuat tabel dengan suatu nama tertentu
- **ALTER** : user berhak untuk mengubah tabel dengan menggunakan pernyataan ALTER TABLE
- **INDEX** : user berhak untuk menentukan indeks pada satu tabel
- **DROP** : user berhak untuk menghapus tabel
- **ALL** atau **ALL PRIVILEGES** : merupakan 'singkatan' atas seluruh hak akses yang telah disebutkan di atas

# Hak Akses :Tabel & Kolom (contd-2)

- **Contoh 5:** Berikanlah hak SELECT untuk JUN1 pada tabel PLAYERS!

**GRANT SELECT ON PLAYERS TO JUN1**

```
GRANT SELECT ON tennis.players TO 'JUN1'@'localhost';
```

- **Contoh 6:** Berikanlah hak SELECT untuk user baru BOB pada tabel PLAYERS!

**GRANT SELECT ON PLAYERS TO 'BOB'@'localhost' IDENTIFIED BY 'BOB\_PASS'**

```
GRANT SELECT ON PLAYERS TO 'BOB'@'localhost' IDENTIFIED BY 'BOB_PASS'
```

- **Contoh 7:** Berikanlah hak INSERT dan UPDATE untuk EVE1 dan JUN1 pada seluruh kolom dari tabel TEAMS!

**GRANT INSERT, UPDATE ON TEAMS TO EVE1, JUN1**

```
GRANT insert,update,ALTER ON tennis.teams TO  
'JUN1'@'localhost';
```

# Pemberian Akses DBA ke User account terhadap database:

- Membatasi User akses data **table** baik untuk **melihat struktur table, melihat data** maupun melakukan **operasi manipulasi data** seperti **Insert, Update, Delete** data
- Membatasi user akses **view** database baik melihat, maupun merubah struktur **view**
- Membatasi user akses **strored procedure** baik **execute** dan merubah struktur **SQL** didalam **stored procedure** tersebut.
- Membatasi Host akses yang digunakan user baik local host(**127.0.0.1**), user akses dalam **jaringan LAN** dan Remot IP Public.
- Memberikan **timer user** akses **database** pada saat **jam kerja** saja misalnya diluar jam kerja user tidak dapat melakukan akses database.
- Memberikan **otoritas user** untuk **create tabel, view function, stored procedure , trigger**

# Contoh Perintah SQL

- ❑ **GRANT** : memberikan wewenang kepada pemakai
- ❑ Syntax : GRANT ON TO
- ❑ Contoh :
  - ❑ GRANT SELECT ON S TO BUDI
  - ❑ GRANT SELECT,UPDATE (STATUS,KOTA) ON S TO SULIS,WENDI
- ❑ **REVOKE** : mencabut wewenang yang dimiliki oleh pemakai
- ❑ Syntax : REVOKE ON FROM
- ❑ Contoh :
  - ❑ REVOKE SELECT ON S TO BUDI
  - ❑ REVOKE SELECT,UPDATE (STATUS,KOTA) ON S TO SULIS,WENDI
- ❑ **Priviledge list** : READ, INSERT, DROP, DELETE, INEX, ALTERATION, RESOURCE

# Hak Akses : Basisdata

- **SELECT** — user berhak untuk **mengakses seluruh tabel dan view** dari suatu basisdata dengan menggunakan pernyataan **SELECT**
- **INSERT** — user berhak untuk **menambah baris pada seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **INSERT**
- **DELETE** — user berhak untuk **menghapus baris dari seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **DELETE**
- **UPDATE** — user berhak untuk **mengubah nilai dari seluruh tabel** yang ada dalam suatu basisdata dengan menggunakan pernyataan **UPDATE**
- **REFERENCES** — user berhak untuk membuat **Foreign Key ( FK)** yang mengacu pada tabel-tabel yang ada dalam suatu basisdata
- **CREATE** — **CREATE** — user berhak untuk **membuat tabel-tabel baru** dalam suatu basisdata dengan menggunakan pernyataan **CREATE TABLE**
- **ALTER** — user berhak untuk **mengubah tabel-tabel** pada suatu basisdata dengan menggunakan pernyataan **ALTER TABLE**

# Hak Akses : Basisdata (contd-2)

- **DROP** —user berhak untuk **menghapus seluruh tabel dan view** yang ada dalam suatu basisdata
- **INDEX** —user berhak untuk **menentukan dan menghapus indeks** dari tabel-tabel yang ada dalam suatu basisdata
- **CREATE TEMPORARY TABLES** —user berhak untuk **membuat tabel-tabel sementara** di dalam suatu basisdata
- **CREATE VIEW** — user berhak untuk **membuat view baru** dalam suatu basisdata dengan menggunakan pernyataan **CREATE VIEW**
- **SHOW VIEW** — user berhak untuk melihat **definisi view** dari **seluruh view** yang ada dalam suatu basisdata dengan menggunakan pernyataan **SHOW VIEW**
- **CREATE ROUTINE** —user berhak untuk **membuat prosedur (stored procedure) dan fungsi (stored function)** baru untuk suatu basisdata
- **ALTER ROUTINE** —user berhak untuk **mengubah dan menghapus prosedur (stored procedure) dan fungsi (stored function)** dari suatu basisdata

# Hak Akses : Basisdata (contd-3)

- ❑ **EXECUTE ROUTINE** — user berhak untuk membatalkan prosedur (**stored procedure**) dan fungsi (**stored function**) yang ada pada suatu basisdata
- ❑ **LOCK TABLES** — user berhak untuk **memblok tabel-tabel** yang ada dalam suatu basisdata
- ❑ **ALL** atau **ALL PRIVILEGES** —merupakan 'singkatan' atas seluruh hak akses yang telah disebutkan sebelumnya

# Hak Akses:Basisdata (contd-4)

- **Contoh 8:** Berikanlah hak SELECT untuk JUN1 pada seluruh tabel Yang ada di dalam basisdataTENNIS!

GRANT SELECT ON TENNIS.\* TO JUN1

`GRANT SELECT ON TENNIS.* TO 'JUN1'@'localhost';`

- **Contoh 9:** Berikanlah hak CREATE, UPDATE dan REMOVE tabel-tabel dan view baru untuk JUN1 didalam basisdata TENNIS!

GRANT CREATE, ALTER, DROP, CREATE VIEW ON TENNIS.\* TO JUN1

`GRANT CREATE, ALTER, DROP, CREATE VIEW ON TENNIS.* TO 'JUN1'@'localhost'`

# Hak Akses:User

- Contoh 10: Berikanlah hak CREATE,ALTER,dan DROP untuk EVE1 pada seluruh tabel dari seluruh basisdata!

GRANT CREATE, ALTER, DROP ON \*.\* TO EVE1

**GRANT CREATE, ALTER, DROP ON \*.\* TO EVE1 @'localhost'**

- Contoh 11: Berikanlah hak kepada JUN1 untuk membuat user baru!

GRANT CREATE USER ON \*.\* TO JUN1

**GRANT CREATE USER ON \*.\* TO JUN1@localhost**

# Meneruskan Hak Akses: WITH GRANTOPTION

- Contoh 12: Berikanlah hak REFERENCES untuk JUN1 pada tabel TEAMS dan berikanlah ijin padanya untuk meneruskan hak tersebut kepada user lain!

GRANT REFERENCES ON TEAMS TO JUN1 WITH GRANT OPTION

**GRANT REFERENCES ON TEAMS TO JUN1@localhost WITH GRANT OPTION**

- Dengan adanya klausa WITH GRANT OPTION, maka JUN1 dapat meneruskan hak akses REFERENCES tersebut kepada EVE1

GRANT REFERENCES ON TEAMS TO EVE1

**GRANT REFERENCES ON TEAMS TO EVE1@localhost**

# Membatalkan Hak Akses

- Contoh 13: Batalkan hak SELECT untuk JUN1 pada tabel PLAYERS!

REVOKE SELECT ON PLAYERS FROM JUN1

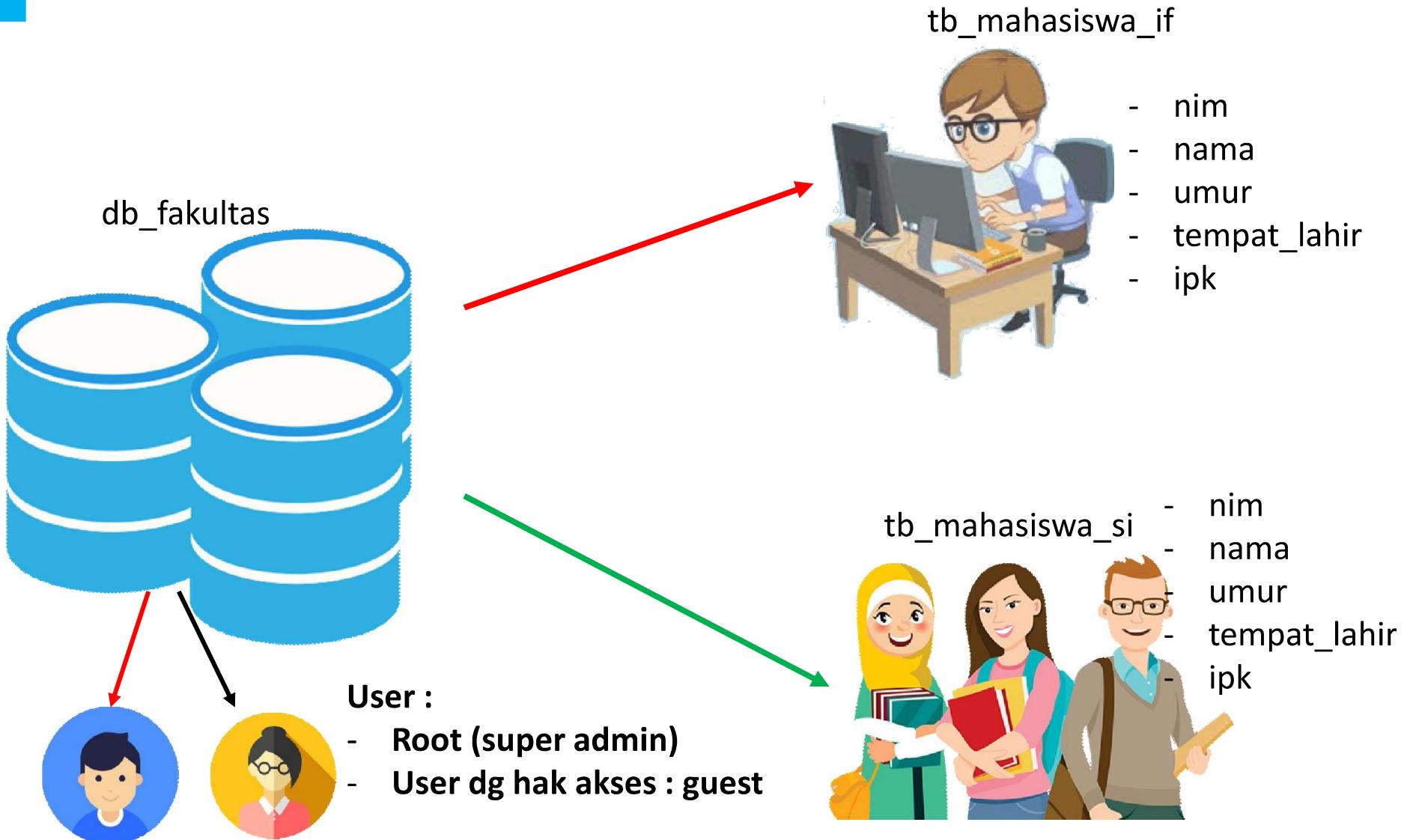
`REVOKE SELECT ON PLAYERS FROM JUN1@localhost`

- Contoh 14: Batalkan hak INSERT dan SELECT untuk EVE1 pada tabel TEAMS!

REVOKE INSERT, SELECT ON TEAMS FROM EVE1

`REVOKE INSERT, SELECT ON teams FROM EVE1@localhost`

# User dan Password



# **SQL - Structured Language Query**

# SQL - Structured Language Query

Adalah bahasa standar yang digunakan untuk **memanipulasi basisdata relasional**

□ Terdiri dari:

□ **Data Definition Language (DDL):**

□ CREATE tables, indexes, views, Establish primary / foreign keys, DROP / ALTER tables .... etc

□ **Data Manipulation Language (DML):**

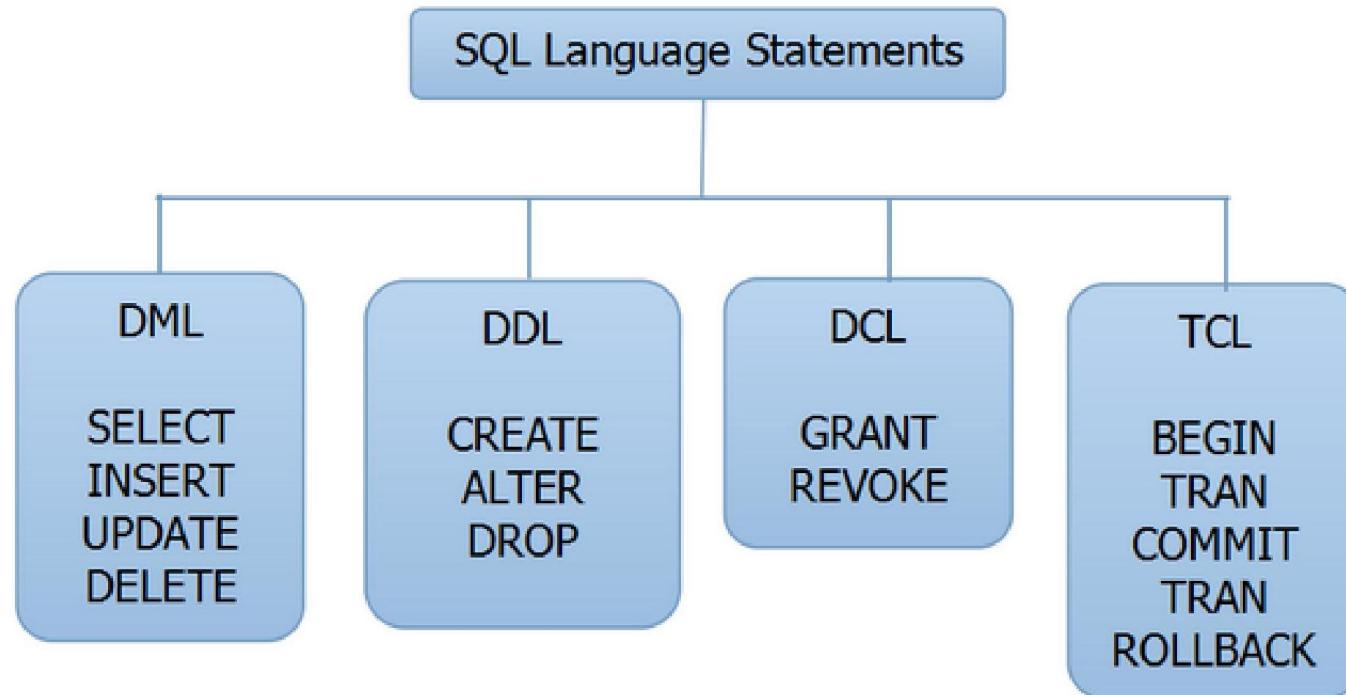
□ INSERT / UPDATE / DELETE, SELECT .... etc.

□ **Data Control Language (DCL):**

□ COMMIT / ROLLBACK work, GRANT / REVOKE .... etc

# SQL

## b) DDL DML

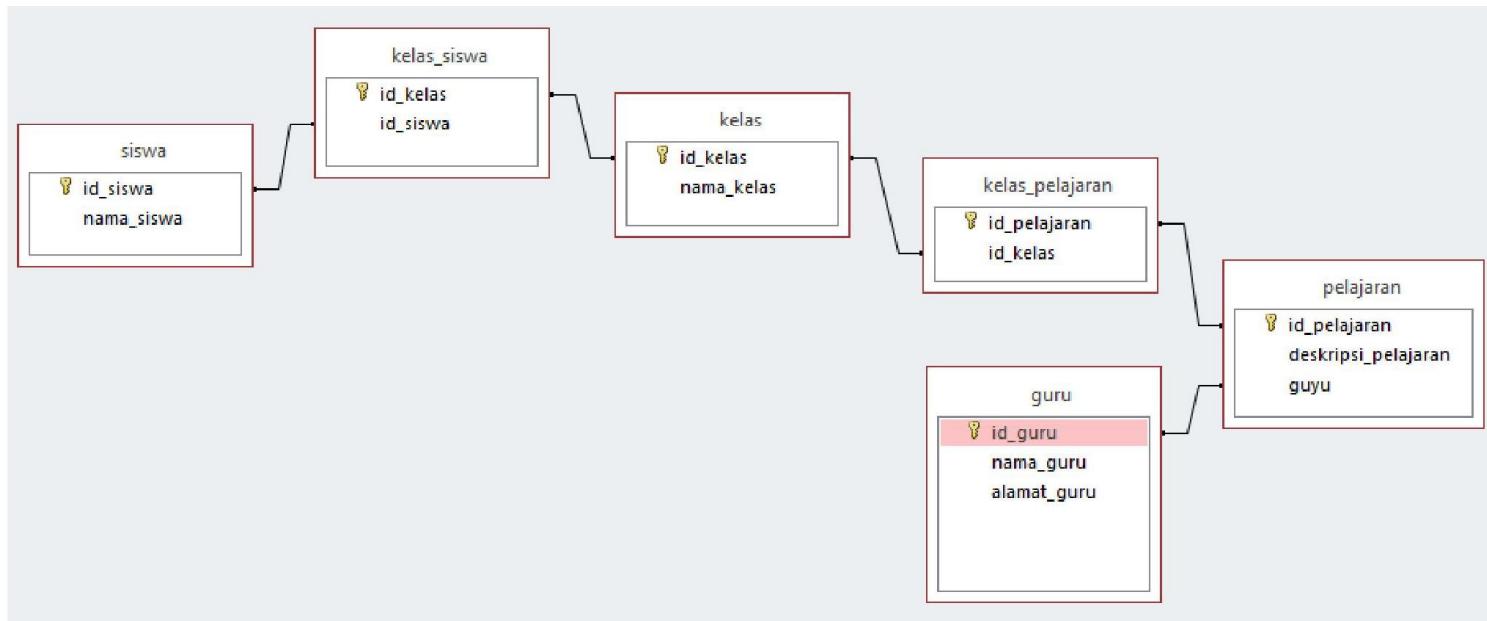


- Data Definition Language (DDL)
- Data Manipulation Language (DML)

# SQL

## b) DDL

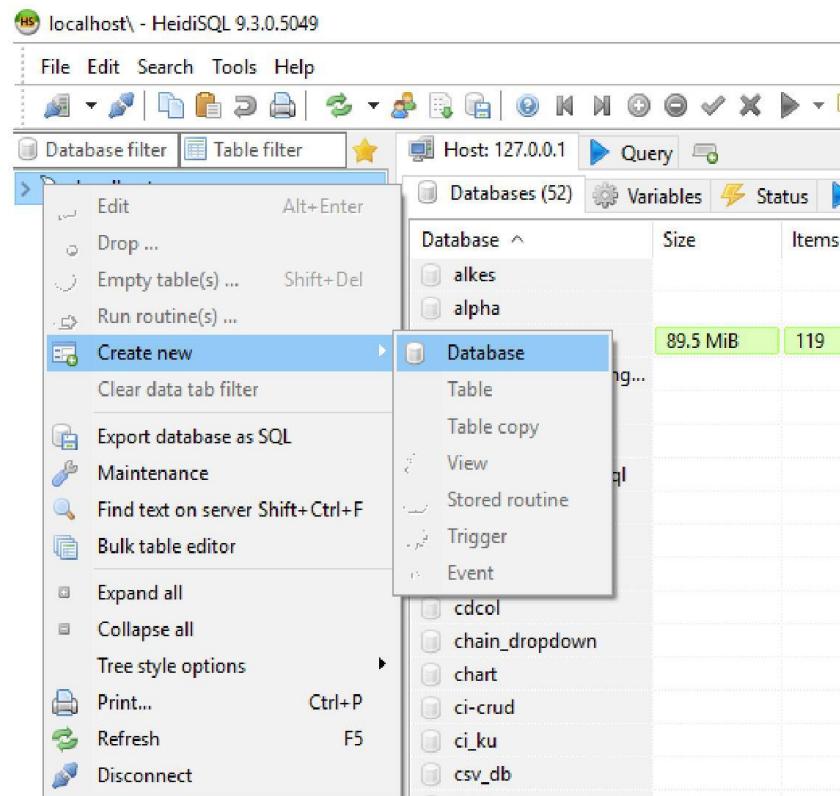
- Pengelolaan pembuatan database dan tabel.
- Dengan berdasar pada relasi table seperti gambar berikut, (nama database = **sekolah**)



# SQL

## b) DDL Script - buat database

GUI



CLI

**CREATE DATABASE**

`sekolah`;

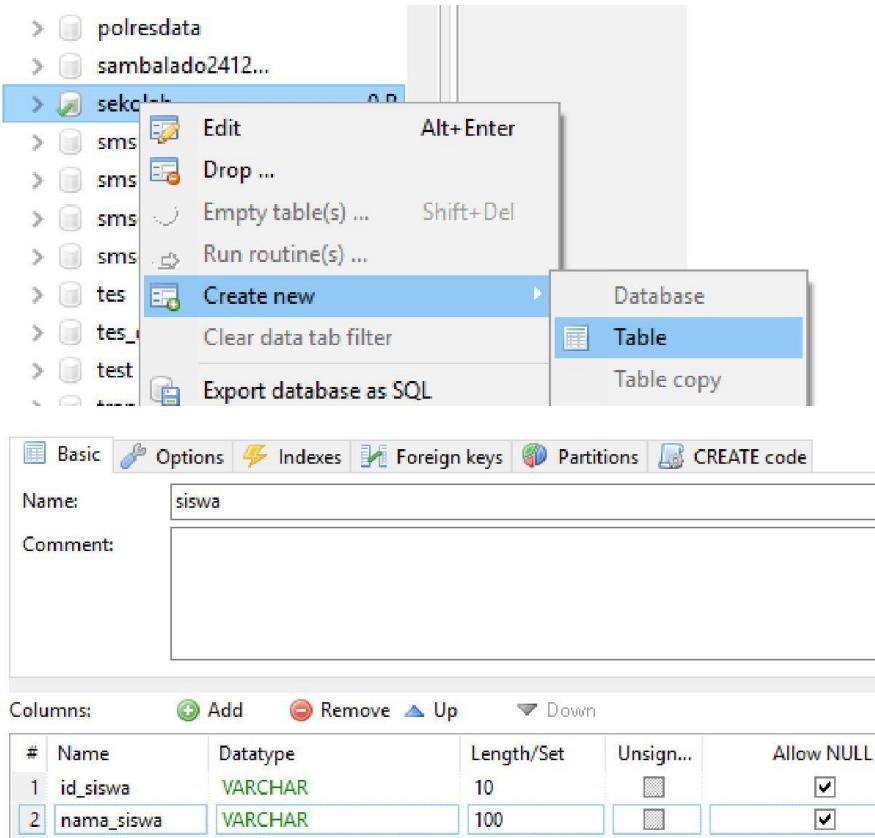
**USE `sekolah`;**

**SHOW TABLES;**

# SQL

## b) DDL Script - buat tabel

GUI



CLI

```
CREATE TABLE `siswa` (
  `id_siswa` VARCHAR(10)
NULL,
  `nama_siswa` VARCHAR(100)
NULL
);
```

```
SHOW TABLES;
```

# Latihan Praktek DDL

# Latihan - *CREATE/DROP*basisdata

- **CREATE DATABASE db\_name**

Contoh:

- CREATE DATABASE tennis
- USE tennis
- CREATE DATABASE president
- USE president

- **DROP DATABASE db\_name**

Contoh:

- DROP DATABASE tennis
- DROP DATABASE president

# Latihan - *CREATE*tabel

```
CREATE TABLE tbl_name (  
    column_name data_type [DEFAULT  
expr]  
    [column_constraint] , ...  
    [table_constraint] );
```

# Latihan - *CREATE* tabel (contd-1)

## Contoh: Buat tabel Players!

Statement SQL:

```
CREATE TABLE PLAYERS  
  (PLAYERNO SMALLINT NOT NULL,  
   NAME CHAR(15) NOT NULL,  
   INITIALS CHAR(3) NOT NULL,  
   BIRTH_DATE DATE ,  
   SEX CHAR(1) NOT NULL,  
   JOINED SMALLINT NOT NULL,  
   STREET CHAR(15) NOT NULL,  
   HOUSENO CHAR(4) ,  
   POSTCODE CHAR(6) ,  
   TOWN CHAR(10) NOT NULL,  
   PHONENO CHAR(10) ,  
   LEAGUENO CHAR(4) ,  
   PRIMARY KEY (PLAYERNO));
```

# Latihan - *CREATE*tabel (contd-2)

**Contoh: Buat tabel Committee\_Members!**

Statement SQL:

```
CREATE TABLE COMMITTEE_MEMBERS  
  (PLAYERNO SMALLINT NOT NULL,  
   BEGIN_DATE DATE NOT NULL,  
   END_DATE DATE ,  
   POSITION CHAR(20) ,  
   PRIMARY KEY (PLAYERNO, BEGIN_DATE) );
```

# Latihan - Skema Tabel (I)

## PLAYERS

PLAYERNO (PLAYERNO) NOT NULL	NAME (NAME) NOT NULL	INITIALS (INITIALS) NOT NULL	BIRTH_DATE (DATE)	SEX (SEXCODE) NOT NULL	JOINED (DATE) NOT NULL
------------------------------------	----------------------------	------------------------------------	----------------------	------------------------------	------------------------------

<—PK—>

STREET (STREETNAME) NOT NULL	HOUSENO (HOUSENO)	POSTCODE (POSTCODE)	TOWN (TOWNNAME) NOT NULL	PHONENO (PHONENO)	LEAGUENO (LEAGUENO)
------------------------------------	----------------------	------------------------	--------------------------------	----------------------	------------------------

<—————>

## TEAMS

TEAMNO (TEAMNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	DIVISION (DIVISIONNAME) NOT NULL
--------------------------------	------------------------------------	--

<—PK—> <—————>

# Latihan - Skema Tabel (2)

## MATCHES

MATCHENO (MATCHENO) NOT NULL	TEAMNO (TEAMNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	WON (NR_OF_SETS) NOT NULL	LOST (NR_OF_SETS) NOT NULL
------------------------------------	--------------------------------	------------------------------------	---------------------------------	----------------------------------

<—PK—>

## PENALTIES

PAYMENTNO (PAYMENTNO) NOT NULL	PLAYERNO (PLAYERNO) NOT NULL	PAYMENT_DATE (DATE) NOT NULL	AMOUNT (AMOUNT) NOT NULL
--------------------------------------	------------------------------------	------------------------------------	--------------------------------

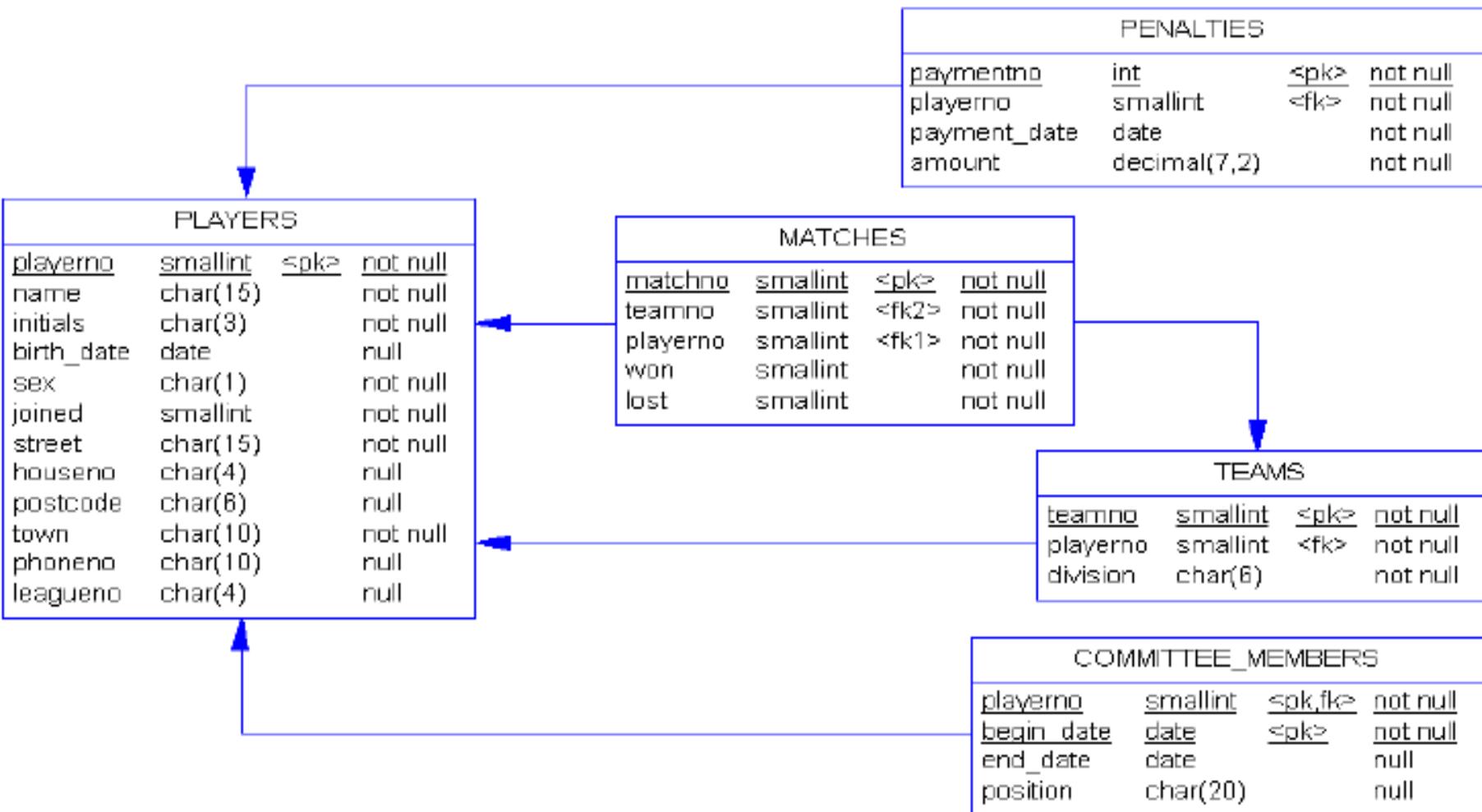
<—PK—>

## COMMITTEE\_MEMBERS

PLAYERNO (PLAYERNO) NOT NULL	BEGIN_DATE (DATE) NOT NULL	END_DATE (DATE)	POSITION (POSITIONNAME) NOT NULL
------------------------------------	----------------------------------	--------------------	--

<————PK————>

# Latihan - Skema Basisdata



# Latihan - Domains & domain constraints

- **PLAYERNO** = NUMERIC (4,0) values  $\geq 1$
- **NAME** = CHARACTER (15)
- **INITIALS** = CHARACTER (3)
- **DATE** = DATE
- **YEARNO** = NUMERIC (4,0)
- **SEXCODE** = CHARACTER (1) set of values = {'M','F'}
- **STREETNAME** = CHARACTER (15)
- **HOUSENO** = CHARACTER (4)
- **POSTCODE** = CHARACTER (6)
- **TOWNNAME** = CHARACTER (10)
- **PHONENO** = CHARACTER (10)
- **LEAGUENO** = CHARACTER (4)
- **TEAMNO** = NUMERIC (2,0) values  $\geq 1$
- **DIVISIONNAME** = CHARACTER (6)
- **MATCHENO** = NUMERIC (4,0) values  $\geq 1$
- **NR\_OF\_SETS** = NUMERIC (1,0) set of values = {0,1,2,3}
- **PAYMENTNO** = NUMERIC (8,0) values  $\geq 1$
- **AMOUNT** = NUMERIC (7,2)
- **POSITIONNAME** = CHARACTER (20) set of values = {'Chairman','Secretary','Treasurer','General member'}

# Latihan - *Intra table constraints*

## Primary keys:

Indicated in the table schema's: <---PK--->

Note: Unique and all columns NOT NULL

## Alternative keys:

Indicated in the table schema's:<----->

Note: Unique, not necessary NOT NULL

## NOT NULL constraints:

Indicated in the table schema's:NOT NULL

## Column value constraints:

PLAYERS (JOINED) values  $\geq$  1970

PENALTIES (PAYMENT\_DATE) values  $\geq$  '1970-01-01'

PENALTIES (AMOUNT) values  $>$  0.00

COMMITTEE\_MEMBERS (BEGIN\_DATE) values  $>$  '1990-01-01'

## Row constraints:

PLAYERS: YEAR (BIRTH\_DATE)  $\leq$  JOINED

COMMITTEE\_MEMBERS: END\_DATE  $\geq$  BEGIN\_DATE

## Other intra table constraints:

Note: Not present in this database.

# Latihan - Inter table constraints

- *Foreign key references:*

TEAMS (PLAYERNO) ---> PLAYERS (PLAYERNO)

MATCHES (TEAMNO) ---> TEAMS (TEAMNO)

MATCHES (PLAYERNO) ---> PLAYERS (PLAYERNO)

PENALTIES (PLAYERNO) ---> PLAYERS (PLAYERNO)

COMMITTEE\_MEMBERS (PLAYERNO) ---> PLAYERS  
(PLAYERNO)

- *Other inter table constraints:*

*For rows of table PLAYERS that are referenced by a foreign key from tables TEAMS, MATCHES and PENALTIES: LEAGUENO IS NOT NULL*

*For table PENALTIES the value of PAYMENT\_DATE must be greater or*

*equal than the value of JOINED in table PLAYERS for the same player.*

# Latihan - CREATE Tabel (contd-3)

- Contoh: Buat tabel Players!
- Statemen SQL:

```
CREATE TABLE PLAYERS
(PLAYERNO      SMALLINT      NOT NULL,
 NAME          CHAR(15)      NOT NULL,
 INITIALS       CHAR(3)      NOT NULL,
 BIRTH_DATE    DATE         ,
 SEX           CHAR(1)      NOT NULL,
 JOINED        SMALLINT      NOT NULL,
 STREET         CHAR(15)      NOT NULL,
 HOUSENO        CHAR(4)      ,
 POSTCODE       CHAR(6)      ,
 TOWN          CHAR(10)      NOT NULL,
 PHONENO        CHAR(10)      ,
 LEAGUENO        CHAR(4)      ,
 PRIMARY KEY   (PLAYERNO)  ,
 CHECK (PLAYERNO >= 1)  ,
 CHECK (SEX IN ('M', 'F'))  ,
 CHECK (JOINED >= 1970)  ,
 CHECK (YEAR (BIRTH_DATE) <= JOINED) );
```

# Latihan - CREATE Tabel (contd-4)

Contoh: Buat tabel TEAMS!

Statemen SQL:

```
CREATE TABLE TEAMS  
  (TEAMNO SMALLINT NOT NULL,  
   PLAYERNO SMALLINT NOT NULL,  
   DIVISION CHAR(6) NOT NULL,  
   PRIMARY KEY (TEAMNO),  
   UNIQUE (PLAYERNO),  
   FOREIGN KEY (PLAYERNO)  
     REFERENCES PLAYERS (PLAYERNO),  
   CHECK (TEAMNO >= 1);
```

# Latihan - CREATE Tabel (contd-5)

Contoh: Buat tabel Committee\_Members!

Statemen SQL:

```
CREATE TABLE COMMITTEE_MEMBERS  
  (PLAYERNO SMALLINT NOT NULL,  
   BEGIN_DATE DATE NOT NULL,  
   END_DATE DATE ,  
   POSITION CHAR(20) ,  
   PRIMARY KEY (PLAYERNO, BEGIN_DATE) ,  
   FOREIGN KEY (PLAYERNO)  
     REFERENCES PLAYERS (PLAYERNO) ,  
   CHECK (POSITION IN ('Chairman', 'Secretary',  
                      'Treasurer', 'General member')),  
   CHECK (BEGIN_DATE >= '1990-01-01'),  
   CHECK (END_DATE >= BEGIN_DATE) );
```

# Latihan - CREATE Tabel (contd-6)

Contoh: Buat tabel TEAMS!

Statemen SQL:

```
CREATE TABLE TEAMS  
  (TEAMNO SMALLINT NOT NULL PRIMARY KEY  
    CHECK (TEAMNO >= 1),  
    PLAYERNO SMALLINT NOT NULL UNIQUE,  
    FOREIGN KEY (PLAYERNO)  
    REFERENCES PLAYERS (PLAYERNO),  
    DIVISION CHAR(6) NOT NULL);
```

# Latihan - Drop Table

**DROP TABLE tbl\_name [, tbl\_name]**

Contoh: Drop seluruh tabel dalam basisdata tennis!

Statemen SQL:

```
DROP TABLE COMMITTEE_MEMBERS;
DROP TABLE PENALTIES;
DROP TABLE MATCHES;
DROP TABLE TEAMS;
DROP TABLE PLAYERS;
```

# Latihan - RENAME Tabel

- **RENAME TABLE tbl\_name TO new\_tbl\_name**  
[**,tbl\_name2 TO new\_tbl\_name2**] ...

**Contoh:** Ubah nama tabel Teams menjadi Groups!

Statemen SQL:

**RENAME TABLE Teams TO Groups**

*RENAME tabel*

# Latihan - ALTER Tabel (contd-1)

- **ALTER TABLE** `tbl_name`  
`alter_specification [, alter_specification] ...`

**alter\_specification:**

- ADD
- CHANGE & MODIFY
- DROP
- RENAME

*ALTER tabel*

# Latihan - ALTER Tabel ADD

**ADD [COLUMN] col\_name column\_definition [FIRST |  
AFTER col\_name ]**

**ADD [COLUMN] (col\_name column\_definition,...)**

**Contoh:** Tambahkan satu kolom bernama TYPE di dalam tabel TEAMS.  
Kolom TYPE ini untuk memberikan identifikasi tim Pria dan Wanita.

Statemen SQL:

```
ALTER TABLE TEAMS  
ADD COLUMN TYPE CHAR (1);
```

**Contoh:** Dengan adanya kolom TYPE, maka (misalkan) tim 2 sebagai tim  
Pria harus di-update.

Statemen:

```
UPDATE TEAMS  
SET TYPE = 'M'  
WHERE TEAMNO = 2;
```

**ALTER tabel : ADD**

# Latihan - ALTER Tabel CHANGE & MODIFY

**CHANGE [COLUMN] old\_col\_name new\_col\_name  
column\_definition [FIRST|AFTER col\_name]**

**MODIFY [COLUMN] col\_name column\_definition  
[FIRST | AFTER col\_name]**

**Contoh:** Tambahkan panjang karakter kolom TOWN  
dari 10 menjadi 20!

```
ALTER TABLE PLAYERS  
MODIFY COLUMN TOWN CHAR (20);
```

*ALTER tabel : CHANGE & MODIFY*

# Latihan - DROP

**DROP [COLUMN] col\_name**

**DROP PRIMARY KEY**

**DROP FOREIGN KEY fk\_symbol**

**Contoh:** Hapuskan kolom TYPE dari tabel TEAMS!

Statemen:

**ALTER TABLE TEAMS**

**DROP COLUMN TYPE**

# Latihan - ALTER Tabel RENAME

**RENAME [TO] new\_tbl\_name**

**Contoh:** Ubah nama tabel Teams menjadi Groups!

Statemen SQL:

**ALTER TABLE Teams RENAME Groups**

*ALTER tabel : RENAME*

# DML (DATA MANIPULATION LANGUAGE)

- 1) *INSERT* data ke dalam tabel
- 2) *SELECT*(dari 1 tabel)
- 3) *SELECT*(lebih dari 1 tabel)
- 4) *UPDATE* data dalam tabel
- 5) *DELETE* data dalam tabel

# SQL

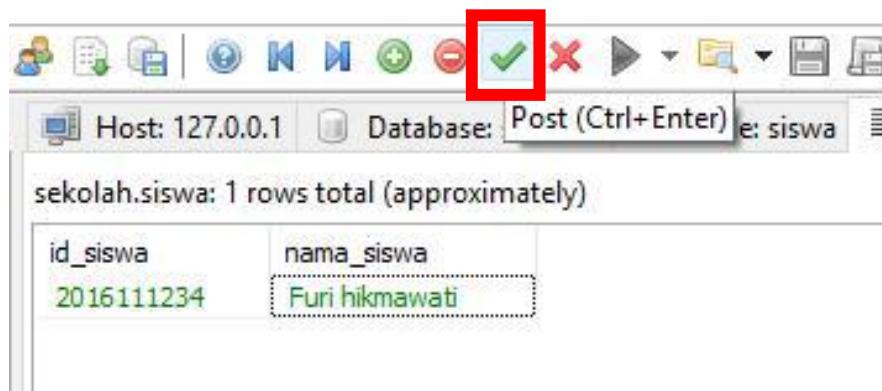
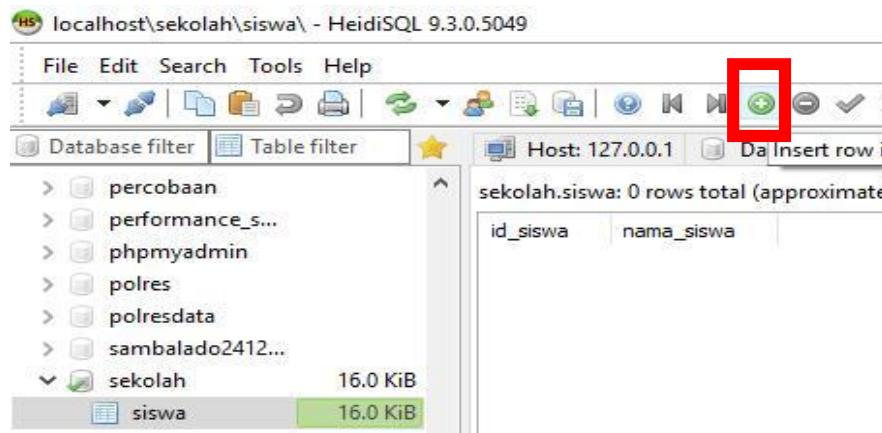
## c) DML

- ❑ Pengelolaan **data** dalam tabel.
- ❑ Bentuk **CRUD**:
  1. **Create**
  2. **Read** (Max, Min, Sum, dll)
  3. **Update**
  4. **Delete**

# SQL

## c) DML Script - Create

GUI



CLI

### INSERT INTO

```
 `sekolah`.`siswa`  
(`id_siswa`,  
 `nama_siswa`) VALUES  
('2016111234', 'Furi  
Hikmawati');
```

# SQL

## c) DML Script - Read

GUI



Host: 127.0.0.1 Database: sekolah Table: siswa Data

sekolah.siswa: 1 rows total (approximately)

id_siswa	nama_siswa
2016111234	Furi hikmawati

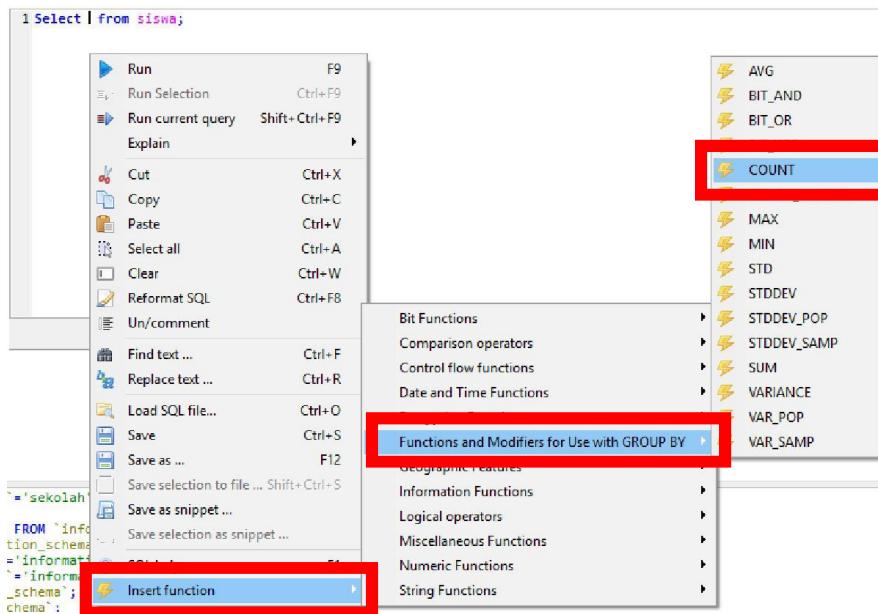
CLI

```
SELECT * FROM  
`sekolah`.`siswa`;
```

# SQL

## c) DML Script - Read (count)

GUI



CLI

```
Select COUNT(*) from siswa;
```

# SQL

## c) DML Script – Read (count)

### Statemen SELECT

Cari nomor pemain yang telah melakukan setidaknya dua kali penalti yang jumlahnya lebih dari \$25! Urutkan hasil berdasarkan nomor pemainnya!

### Query:

```
SELECT PLAYERNO FROM PENALTIES WHERE AMOUNT >  
25 GROUP BY PLAYERNO HAVING COUNT (*) > 1 ORDER  
BY PLAYERNO;
```

# SQL

## c) DML Script – Read (count)

**Query:**

```
SELECT          PLAYERO NO  
FROM           PENALTIES  
WHERE          AMOUNT > 25  
GROUP BY       PLAYERO NO  
HAVING         COUNT (*) > 1  
ORDER BY       PLAYERO NO;
```

**FROM**

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
3	27	1983-09-10	100.00
4	104	1984-12-08	50.00
5	44	1980-12-08	25.00
6	8	1980-12-08	25.00
7	44	1982-12-30	30.00

**WHERE**

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
3	27	1983-09-10	100.00
4	104	1984-12-08	50.00
7	44	1982-12-30	30.00
8	27	1984-11-12	75.00

**GROUP BY**

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
1	6	1980-12-08	100.00
2	44	1981-05-05	75.00
7	44	1982-12-30	30.00
3	27	1983-09-10	100.00
8	27	1984-11-12	75.00
4	104	1984-12-08	50.00

**HAVING**

PAYMENTNO	PLAYERO NO	PAYMENT_DATE	AMOUNT
2	44	1981-05-05	75.00
7	44	1982-12-30	30.00
3	27	1983-09-10	100.00
8	27	1984-11-12	75.00

**SELECT**

PLAYERO NO
44
27

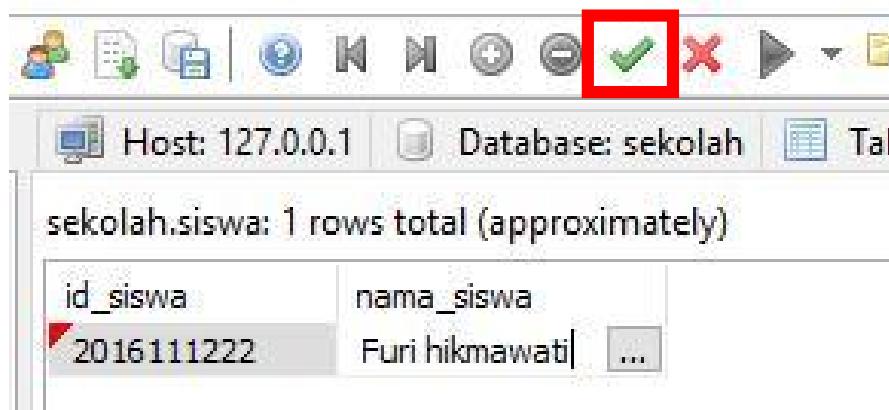
**ORDER BY**

PLAYERO NO
27
44

# SQL

## c) DML Script - Update

GUI



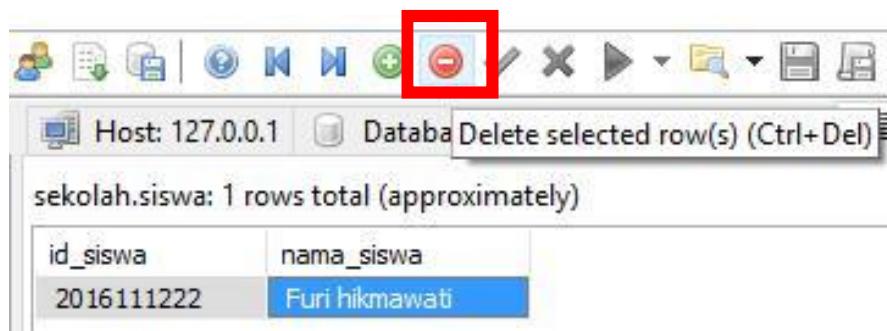
CLI

```
UPDATE `sekolah`.`siswa`
SET
`id_siswa`='2016111222'
WHERE
`id_siswa`='2016111234'
AND `nama_siswa`='Furi
hikmawati' LIMIT 1;
```

# SQL

## c) DML Script - Delete

GUI



CLI

### **DELETE FROM**

```
DELETE FROM
`sekolah`.`siswa` WHERE
`id_siswa`='2016111222'
AND `nama_siswa`='Furi
hikmawati' LIMIT 1;
```

# Start MySQL

```
C:\Users\Smart>
```

```
cd c:\xampp\mysql\bin\
```

```
c:\xampp\mysql\bin>
```

```
mysql -u root -p
```

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Microsoft Windows  
[Version 6.1.7600]

```
MariaDB [(none)]> show databases;
```

```
+-----+
```

```
| Database |
```

```
+-----+
```

```
| fifin |
```

```
| information_schema |
```

```
| mysql |
```

```
| performance_schema |
```

```
| phpmyadmin |
```

```
| test |
```

```
+-----+
```

```
6 rows in set (0.11 sec)
```

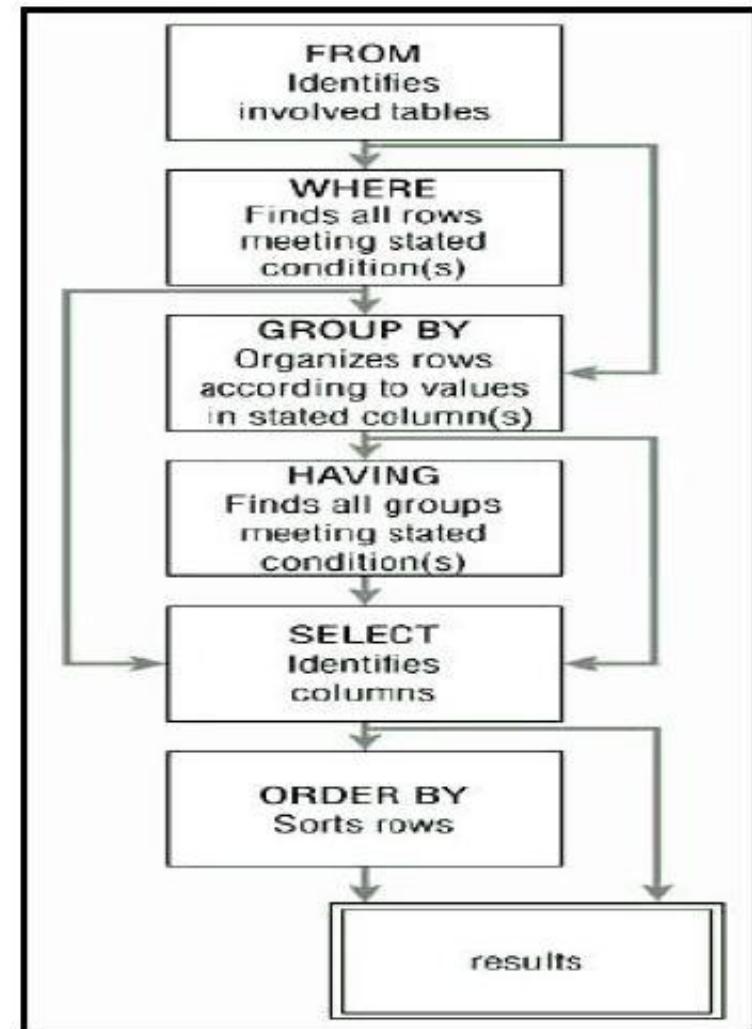
```
MariaDB [(none)]> use fifin;
```

```
Database changed
```

```
MariaDB [fifin]>
```

# *Data Manipulation Language (DML)*

- *INSERT / UPDATE / DELETE, SELECT*
- Klausula-klausula *SELECT*:
  - **FROM** --> menentukan tabel(-tabel) sumber
  - **WHERE** --> memilih baris(-baris) yang memenuhi kondisi (-kondisi)
  - **GROUP BY** --> menggabungkan baris(-baris) yang kolomnya memiliki nilai yang sama
  - **HAVING** --> memilih grup yang memenuhi kondisi tertentu
  - **SELECT** --> memilih kolom(-kolom)
  - **ORDER BY** --> mengurutkan baris-baris berdasarkan nilai-nilai yang ada dalam kolom(-kolom)



# *DML: Insert data ke dalam Tabel*

## 1. INSERT satu baris

**INSERT INTO <table specification>**

**[<column list>]**

**VALUES (<expression> [{,<expression>}... ])**

## 2. INSERT dari tabel lain (Mengisi suatu tabel dari isi tabel lain)

**INSERT INTO<table specification>**

**[<column list>]**

**<select clause>**

**<from clause>**

**[ <where clause> ]**

**[ <group by clause> ]**

**[ <having clause> ]**

# DML: Insert data ke dalam Tabel (contd -2)

**Contoh 1:** Sebuah tim baru dengan pemain nomor 100 sebagai kaptennya telah bergabung di dalam liga. Tim ketiga ini akan bermain di dalam divisi ketiga.

Query:

```
INSERT INTO TEAMS (TEAMNO, PLAYERNO, DIVISION)  
VALUES (3,100,'third');
```

atau:

```
INSERT INTO TEAMS VALUES (3,100,'third');
```

# DML: Insert data dari Tabel lain (contd -3)

- **Contoh 1.1:** Buatlah suatu tabel baru yang mencatat nomor pemain, nama, kota dan nomor telepon dari masing-masing pemain yang tidak memiliki nomor liga (tidak pernah bermain di dalam liga)!

Query:

- Membuat tabel baru:

```
CREATE TABLE RECR_PLAYERS  
(PLAYERNO SMALLINT NOT NULL,  
NAME CHAR(15) NOT NULL,  
TOWN CHAR(10) NOT NULL,  
PHONENO CHAR(10) ,  
PRIMARY KEY (PLAYERNO) )
```

- Insert data ke tabel RECR\_PLAYERS dari tabel PLAYERS:

```
INSERT INTO RECR_PLAYERS  
SELECT PLAYERNO, NAME, TOWN, PHONENO  
FROM PLAYERS  
WHERE LEAGUENO IS NULL
```

# KLAUSA : SELECT (dari 1 tabel)

**Contoh 2:** Tampilkan seluruh data penalti.

*Query:*

```
SELECT * FROM Penalties;
```

**Contoh 3:** Tampilkan nomor pembayaran, nomor pemain dan jumlah dari masing-masing penalti

*Query:*

```
SELECT paymentno, playerno, amount  
FROM Penalties;
```

Contoh 3.1 : Tampilkan nomor pemain, nama dan umur ketika bergabung di klub tenis dari masing-masing pemain.

*Query:*

```
SELECT playerno, name,  
       joined - Year(birth_date) AS join_age  
FROM Players
```

# KLAUSA : SELECT - WHERE (dari 1 tabel)

**Contoh 4:** Dapatkan nomor pemain dan nomor liga dari masing-masing pemain yang memang benar-benar memiliki nomor liga.

*Query:*

```
SELECT playerno, leagueno  
FROM Players  
WHERE leagueno IS NOT NULL;
```

## KLAUSA : SELECT - BETWEEN (dari 1 tabel)

**Contoh 5:** Cari nomor dan tanggal lahir dari masing-masing pemain yang lahir antara tahun 1962 sampai 1964.

**Query:**

```
SELECT playerno, birth_date  
FROM Players  
WHERE Year(birth_date) BETWEEN 1962 AND  
1964;
```

# KLAUSA : SELECT - IN (dari 1 tabel)

**Contoh 6:** Cari nomor, nama dan kota dari masing-masing pemain yang tinggal di *Inglewood*, *Plymouth*, *Midhurst* atau *Douglas*.

**Query dengan OR:**

```
SELECT playerno, name, town  
FROM Players  
WHERE town = 'Inglewood' OR town = 'Plymouth' OR  
town = 'Midhurst' OR town = 'Douglas';
```

**Query dengan IN:**

```
SELECT playerno, name, town  
FROM Players  
WHERE town IN ('Inglewood', 'Plymouth', 'Midhurst', 'Douglas');
```

# KLAUSA : SELECT – LIKE, ORDER BY (dari 1 tabel)

**Contoh 7:** Dapatkan nama dan nomor dari masing-masing pemain yang memiliki huruf “e” pada posisi huruf sebelum huruf terakhir dari namanya.

Query:

```
SELECT name, playerno  
FROM Players  
WHERE name LIKE '%e_';
```

**Contoh 8:** Buat daftar nomor pembayaran dan nomor pemain untuk masing-masing penalti; urutkan hasil berdasarkan nomor pemain dan nomor pembayaran untuk masing-masing pemain.

Query:

```
SELECT paymentno, playerno  
FROM Penalties  
ORDER BY playerno, paymentno;
```

# Seluruh Klausula dalam 1 Statement

- **Contoh 8.1:** Cari nomor dari masing-masing pemain yang telah melakukan lebih dari 1 penalti yang besarnya lebih dari 25.
- Query:

```
SELECT playerno  
FROM Penalties  
WHERE amount > 25  
GROUP BY playerno  
HAVING COUNT(*) > 1  
ORDER BY playerno
```

## KLAUSA : SELECT (dari 1 tabel)

### Fungsi dalam Select: Count

**Contoh 9:** Hitung jumlah pemain yang tercatat di dalam tabel PLAYERS!

**SQL:**

```
SELECT COUNT(*) FROM PLAYERS;
```

**Contoh 10:** Ada berapa nama kota yang tercatat di dalam kolom TOWN dalam tabel PLAYERS?

**SQL:**

```
SELECT COUNT(DISTINCT(TOWN)) FROM PLAYERS;
```

KLAUSA : SELECT (dari 1 tabel)

*Fungsi dalam Select: Max & Min*

**Contoh 11:** Berapakah jumlah penalti yang tertinggi?

**SQL:**

```
SELECT MAX(AMOUNT) FROM PENALTIES;
```

**Contoh 12:** Berapakah jumlah penalti yang terendah?

**SQL:**

```
SELECT MIN(AMOUNT) FROM PENALTIES;
```

## KLAUSA : SELECT (dari 1 tabel)

*Fungsi dalam Select: Sum*

**Contoh 13:** Berapa banyak total SET yang telah dimenangkan, total SET yang telah kalah, dan berapa perbedaan di antara keduanya?

**SQL:**

```
SELECT SUM(WON),SUM(LOST),SUM(WON)-  
SUM(LOST) AS Difference
```

```
FROM MATCHES;
```

## KLAUSA : SELECT (dari 1 tabel)

*Fungsi dalam Select: Avg*

**Contoh 14:** Berapakah jumlah rata-rata penalti yang dilakukan oleh pemain nomor 44?

*SQL:*

```
SELECT AVG(AMOUNT)  
FROM PENALTIES  
WHERE PLAYERNO = 44;
```

## KLAUSA : SELECT (lebih dari 1 tabel)

**Contoh 17:** Cari nama dan inisial pemain yang telah bermain di dalam pertandingan minimal sebanyak 1 kali!

Query:

```
SELECT DISTINCT P.NAME, P.INITIALS  
FROM PLAYERS AS P, MATCHES AS M  
WHERE P.PLAYERNO = M.PLAYERNO;
```

## KLAUSA : SELECT (lebih dari 1 tabel)

**Contoh 18:** Cari nomor dan nama pemain yang tinggal di kota yang sama dengan pemain nomor 27!

*Query:*

**SELECT P.PLAYERNO, P.NAME**

**FROM PLAYERS AS P, PLAYERS AS P27**

**WHERE P.TOWN = P27.TOWN**

**AND P27.PLAYERNO = 27**

**AND P.PLAYERNO <> 27**

# UPDATE data dalam Tabel

```
UPDATE <table specification>
    SET <column name = expression>
        [ WHERE <condition> ]
```

**Contoh 19:** Keluarga Parmenter telah pindah rumah. Sekarang mereka tinggal di Palmer Street nomor 83, kota Inglewood, kode pos 1234UU. Nomor telepon mereka yang baru belum diketahui.

*Query:*

```
UPDATE PLAYERS
    SET STREET = 'Palmer Street', HOUSENO = '83',
        TOWN = 'Inglewood', POSTCODE = '1234UU',
        PHONENO = NULL
    WHERE NAME = 'Parmenter';
```

# UPDATE data dalam Tabel

**Contoh 20:** Naikkan jumlah penalti sebanyak 5%!

*Query:*

```
UPDATE PENALTIES  
SET AMOUNT = AMOUNT * 1.05;
```

# *DELETE* data dalam Tabel

```
DELETE  
FROM <table specification>  
[ WHERE <condition> ]
```

**Contoh 21:** Hapus seluruh data penalti yang dilakukan oleh pemain nomor 44 pada tahun 1980!

Query:

```
DELETE  
FROM PENALTIES  
WHERE PLAYERNO = 44  
AND YEAR(PAYMENT_DATE) = 1980;
```

## 8) Proyek Akhir

# Proyek Akhir

- Membuat aplikasi sederhana dengan fokus **Penerapan Database** ke Aplikasi untuk menyimpan transaksi
- **Tahapannya :**
  - Penentuan Studi Kasus
  - Perancangan Database beserta Relasi Tabelnya
  - Pada database terdapat beberapa SQL Langguage yang dilakukan diantaranya : CRUD, Transactions, Function, Stored Procedure & Trigger, System Catalog hingga hak akses.
  - Untuk Aplikasi boleh Web atau Desktop, fokus pada penerapan Database.
  - Pembuatan Laporan atau Dokumentasi.
- **Poin penilaian:** Aplikasi (Penerapan Database), Dokumentasi, Presentasi.

# 9) Kebutuhan Software

# Kebutuhan Software

## Browser

- Adobe flash
- Chrome
- Firefox

## Localserver

- Xampp
- Laragon

## Desain Tools

- Power Designer
- Sparx Enterprise Architect

## Editor

- Notepad++
- Sublime Text

## Database GUI

- PostgreSQL
- HeidiSQL
- SQLYog
- FlySpeed SQL

## Database

- Mysql
- Oracle

# 10) Contact

# Contact

- ❑ Bahan Kuliah : [github.com/doniaft](https://github.com/doniaft)
- ❑ Email : [doniaft@gmail.com](mailto:doniaft@gmail.com)
- ❑ WA/Telegram : 0856 4868 8506
- ❑ Komting TIF 4D : Ali Mustofa : 081 231 202 253

## II) Referensi

# Referensi (I)

- ❑ SQL For MySQL Developers, Rick F. van der Lans, Addison Wesley, 2007
- ❑ MySQL Reference Manual, MySQL 2003
- ❑ Database Systems - A Practical Approach to Design, Implementation, and Management, Thomas Connoly and Carolyn Begg, Addison Wesley 1999
- ❑ Raghu Ramakhrisnan, Johannes Gehrke , “Database Management System” 6<sup>rd</sup> Edition, Mc Graw Hill, 2003 - 2006
- ❑ Arief, Rudiyanto M, Pemrograman Basis Data menggunakan Transact-SQL dengan Microsoft SQL Server 2000, Andi Yogyakarta.
- ❑ Rick van der Lans, Introduction to SQL, Mastering Relational Database Language 2nd Edition, Addison-Wesley, 2000.
- ❑ Chris Bates, Web Programming: Building Internet Applications, Third Edition, John Wiley & Sons Ltd, England, 2006.
- ❑ Sebesta, R.W., Programming the World Wide Web, Addison Wesley, 2002.
- ❑ Elliot White III, Jonathan Eisenhamer, PHP 5 in Practice, Sams, 2006
- ❑ Elmasri and Navathe, Fundamental of Database Systems 4th Edition, Addison-Wesley, 2004
- ❑ Silberschatz, Korth and Sudarshan, Database System Concepts, 5th Edition, Mc Graw Hill, International Edition, 2006.
- ❑ Connoly, Thomas and Begg, Carolyn: Database Sytems 4th edition, Prentice Hall, 2005

# Referensi (2)

- Andrea Tar. 2012. PHP and MySQL 24-Hour Trainer
- Brett McLaughlin. 2012. PHP & MySQL- The Missing Manual. USA-Brett McLaughlin. USA-O'REILLY Media
- Brett McLaughlin. 2013. PHP & MySQL- The Missing Manual, 2nd Edition. USA-Brett McLaughlin. USA-O'REILLY Media
- Head First PHP & MySQL
- Kroenke, David. 2013. Database Processing 12th Edition
- Mysql Official. 2016. MySQL 5.7 Reference Manual-en
- PHP6 and MySQL Bible by Steve Suehring
- Rochkin Mark. 2013. Expert PHP and MySQL
- Ruehning, dkk. php\_mysql\_javascript\_\_html5\_all-in-one\_for\_dummies
- Sams.Sams.Teach.Yourself.PHP.MySQL.and.Apache.All-in-One.ISBN0672326205
- Solichin, Achmad. Pemrograman Web dengan PHP MySQL
- Tutorialpoints.com - mysql tutorial
- Valade, Janet. PHP & MySQL Web Development All-in-One Desk Reference For Dummies. CanadaWiley Publishing, Inc
- Widigdo, Anon Kuncoro. 2003. php dan mysql
- <https://www.duniaIlkom.com/tutorial-belajar-mysql-dan-index-artikel-mysql/#mysqllanjutan>