



# **Pengantar Basis Data :**

## **Database, DBMS, RDBMS, Sistem Basis Data**

**PJJ-A11-CF 1234  
Konsep Basis Data  
(3 SKS)**

**Disusun oleh :**

**TIM PENGAMPU MK BASIS DATA**

**URL : [www.kulino.dinus.ac.id](http://www.kulino.dinus.ac.id)**

# Pokok Bahasan (?)

- Apa yang dimaksud Basis Data (**Database**) ?
- Apa yang dimaksud **DBMS**, khususnya **RDBMS** ?
- Mengapa hrs. menggunakan **RDBMS** utk. mengelola data ?
- Bagaimana data aplikasi disajikan dalam **RDBMS** ?
- Bagaimana data dalam **RDBMS** hrs. dicari (*retrieved*) dan dimanipulasi ?
- Bagaimana **RDBMS** mendukung "*concurrent access*" dan memproteksi data selama sistem mengalami kegagalan ?
- Apa komponen-komponen utama dari **RDBMS** ?
- Siapa yang hrs. dilibatkan dengan basis data dalam kehidupan nyata ?

# Capaian Pembelajaran- CP (?)

- Mahasiswa dapat menjelaskan konsep dari basis data dan istilah yang termasuk di dalamnya.
- Mahasiswa dapat menjelaskan keuntungan dan kerugian apabila menggunakan file manajemen basis data
- Mahasiswa dapat menjelaskan konsep data independence, komponen DBMS, fungsi DBMS serta bahasa yang digunakan di dalam DBMS
- Mahasiswa dapat menjelaskan perbedaan model data berbasis objek, record, konseptual dan fisik
- Mahasiswa dapat menyebutkan tugas para pengguna basis data.
- Mahasiswa dapat menjelaskan perbedaan arsitektur dari DBMS.

# Basis Data : Apa yang dimaksud (?)

- **Basis Data/Database : Analogi secara simpel** → Gudang data, Kumpulan data. Pada dasarnya merupakan sistem yang menyimpan record-record data yang ter-komputerisasi.
  - Merupakan sekumpulan *persisten data* yang digunakan oleh sistem aplikasi.
  - *Persisten*, bahwa sekali data diterima oleh (DBMS-Database Management System) untuk disimpan dalam basis data, maka data tersebut hanya dapat dihapus oleh basis data dengan menggunakan permintaan eksplisit kepada DBMS.
- Himpunan kelompok data (*arsip*) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali secara cepat dan mudah.
- Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
- Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.
- **Relation-Database Management System (RDBMS)** : Perangkat lunak yang mendefinisikan database, menyimpan data, mendukung bahasa kueri, menghasilkan laporan, dan membuat layar entri data (*Oracle, DB2, Ms-Sql, My Sql, PostgreSQL, dll*).



# Contoh Basis Data

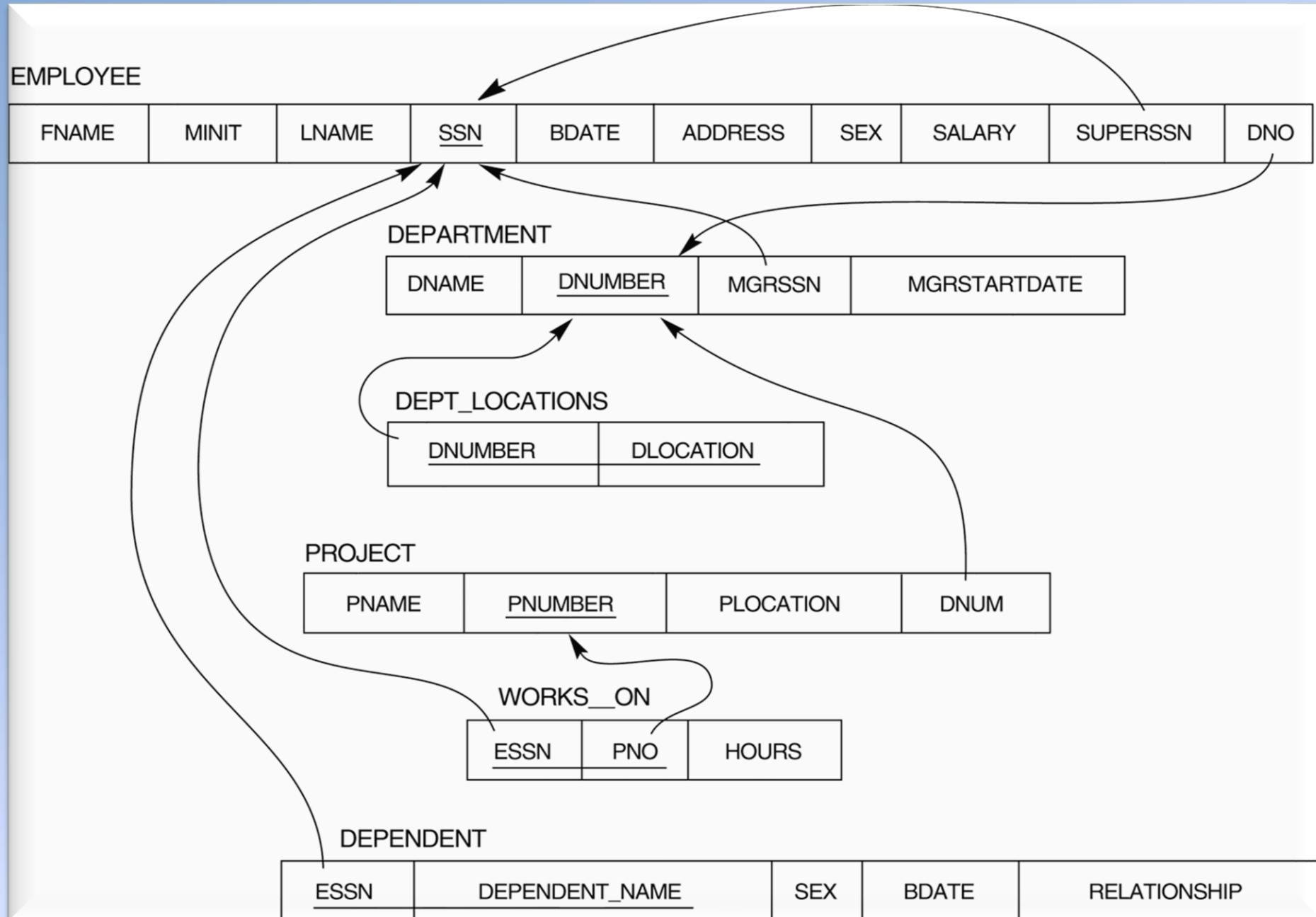
STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Data Structures	CS3320	4	CS
	Discrete Math.	MATH2410	3	MATH

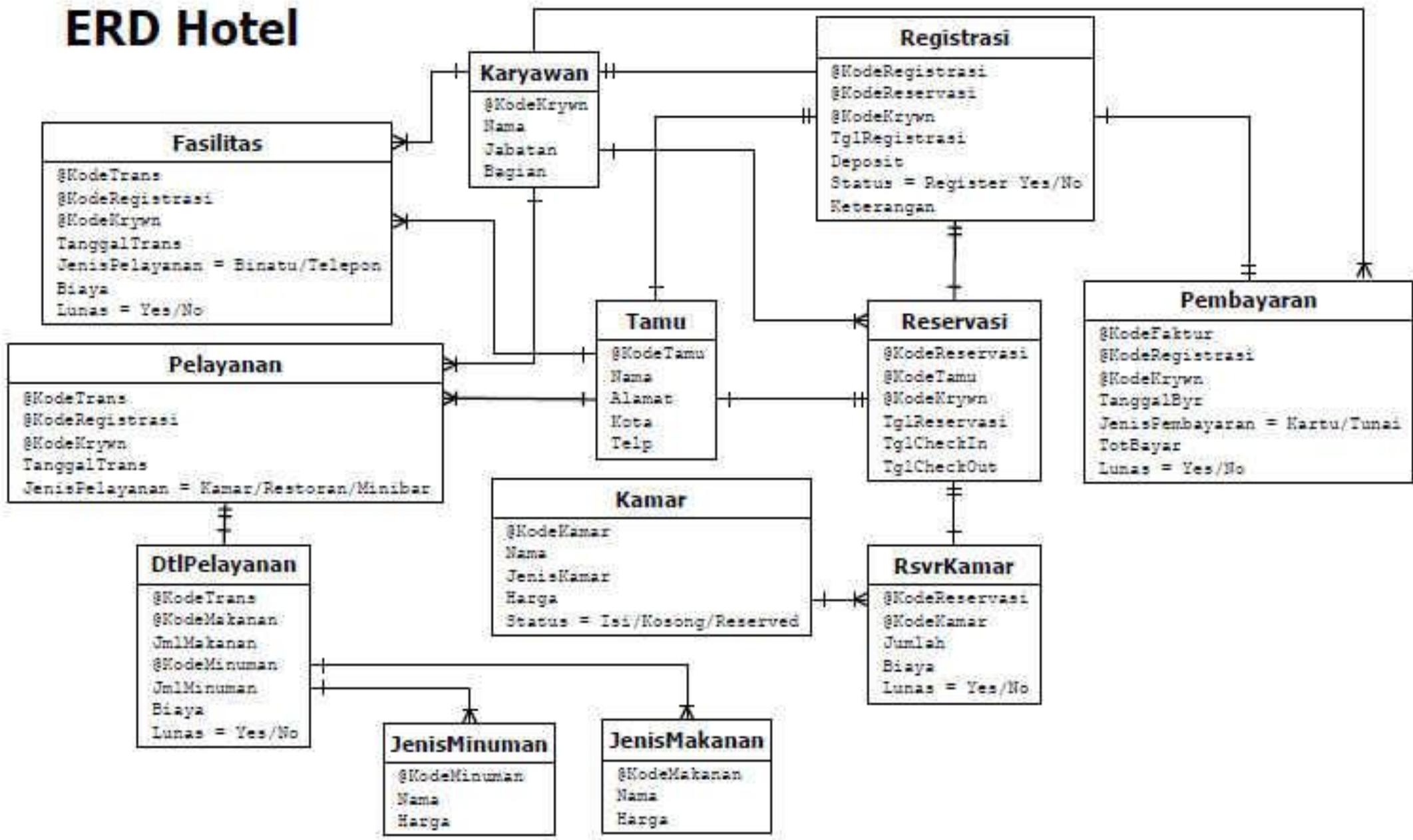
SECTION	SectionID	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	112	CS3320	Fall	98	Anderson

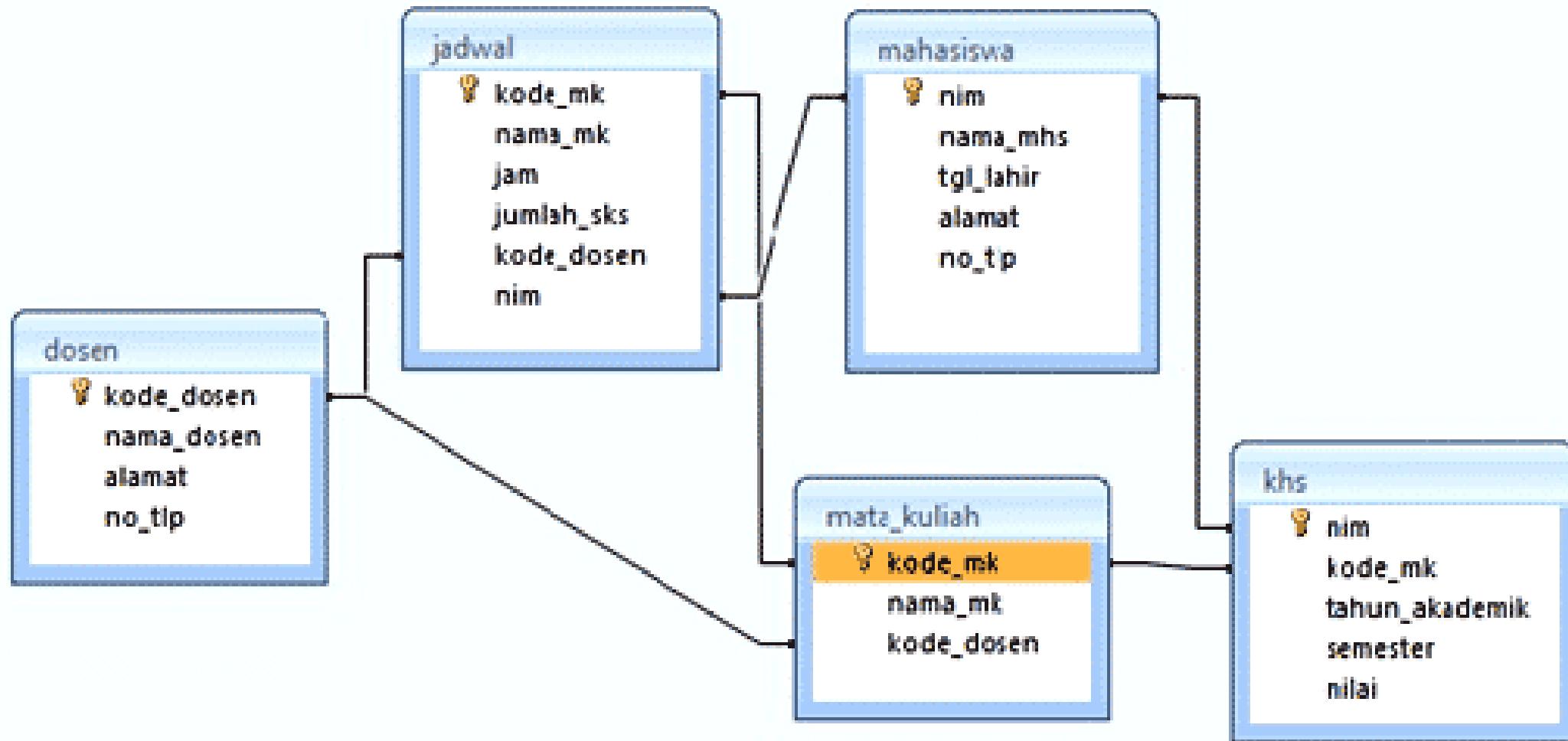
GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	8	85	A

PREREQUISITE	CourseNumber	Prereq_Number
	CS3380	CS3320
	CS3320	CS1310



# ERD Hotel



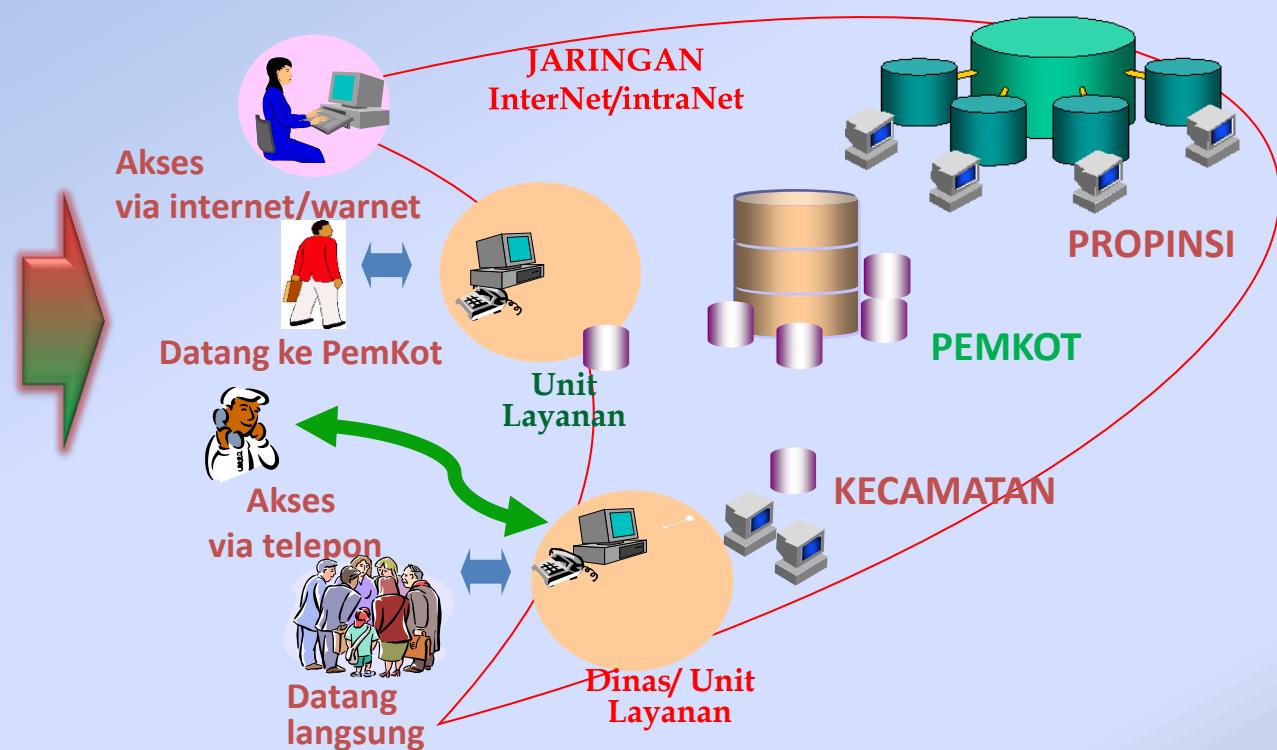
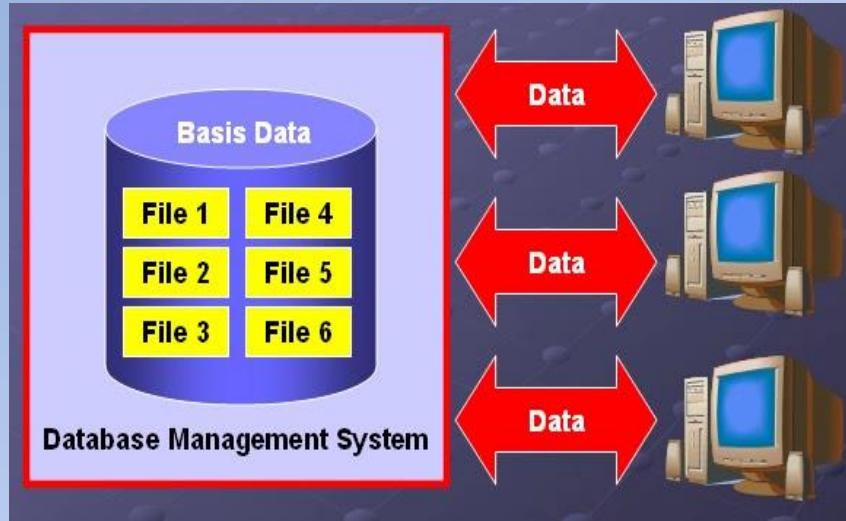


# Basis Data : Maksud & Tujuan (?)

- **Basis Data (Database)**, dapat dibayangkan sebagai sebuah lemari arsip. Jika kita memiliki lemari arsip dan berwenang/bertugas untuk mengelolanya, maka kemungkinan besar kita akan melakukan hal-hal seperti : memberi sampul/map pada kumpulan/bundel arsip yang akan disimpan, menentukan kelompok/jenis arsip, memberi nomor dengan pola tertentu yang nilainya unik pada setiap sampul/map.
- **Basis Data (Database)** : adalah sekumpulan data terintegrasi dengan ukuran yang sangat besar, **dikelola (diolah)** dengan cara tertentu yang secara khusus menjelaskan aktifitas - aktifitas dari satu atau beberapa organisasi yang satu sama lain saling terkait.
- Basis data memodelkan “**dunia nyata**” yang berkaitan dengan :
  - ✓ **Entities** (contoh: mahasiswa, matakuliah, dosen, dlsb.)
  - ✓ **Relationships** (contoh: Madona mengambil matakuliah Basis Data)
- **Secara Umum Tujuan Database ?** : Untuk memudahkan pengeloaan data (menyimpan & mengambil serta mengubah data sesuai kebutuhan secara cepat & tepat).

# Basis Data : Sebagai Sistem (?)

- **Sistem Basis Data :** Sistem yang terdiri atas sekumpulan tabel data yang saling berhubungan dan sekumpulan program ( DBMS : Database Management System) yang memungkinkan berbagai user dan/atau program lain dapat mengakses dan memanipulasi tabel-tabel tersebut.



# Basis Data : Komponen Sistem Basis Data (?)

## Komponen Sistem Basis Data :

- Perangkat Keras (Hardware)  
Komputer, memori, storage (Harddisk), peripheral, dll.
- Sistem Operasi (Operating System)  
Program yang menjalankan sistem komputer, mengendalikan resource komputer dan melakukan berbagai operasi dasar sistem komputer.
- Basis Data (Database)  
Menyimpan berbagai obyek database (struktur tabel, indeks,dll)
- DBMS (Database Management System)  
Perangkat lunak yang memaintain data dalam jumlah besar.
- Pemakai (User)  
Para pemakai database.
- Aplikasi (perangkat lunak) lain.  
Program lain dalam DBMS.

# Database : Apa Tujuannya (?)

Selain untuk kemudahan dalam pengelolaan data, tujuan Database diantaranya adalah untuk meningkatkan :

- *Speed* : Kecepatan pengaksesan serta pengelolaan data.
- *Space* : Efisiensi ruang penyimpanan, berkat adanya anti redundancy.
- *Accuracy* : Akurasi dalam pengelolaan data, berkat adanya struktur & relasi tabel.
- *Availability* : Ketersediaan data.
- *Completeness*: Kelengkapan & Standarisasi Data.
- *Security* : Keamanan data.
- *Sharability* : Kebersamaan pemakaian-Multiuser.

**Tidak Terdapat Pada File System !**

# **FILE MANAGEMENT SYSTEM VS DATA BASE MANAGEMENT SISTEM**

## **FILE MANAGEMENT SYSTEM**

- ❖ PROGRAM ORIENTED
- ❖ KAKU
- ❖ REDUNDANCY DAN INCONSISTENCY

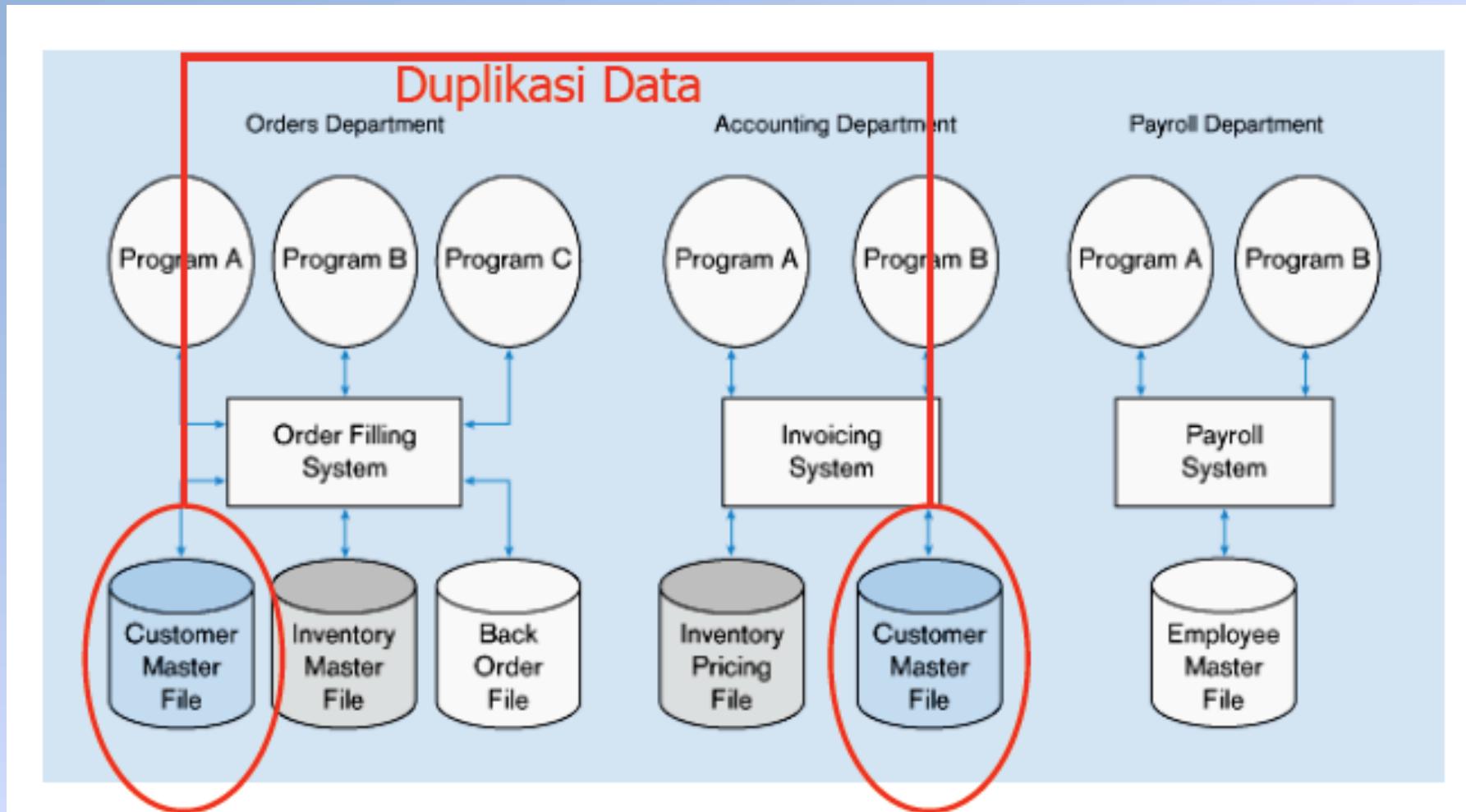
**VS**

## **DATA BASE MANAGEMENT SISTEM**

DATA ORIENTED  
LUWES/FLEKSIBEL  
KESELARASAN DATA TERKONTROL

# File Management System (System File / Sistem Berkas) :

*“sekelompok rekaman disimpan pada sejumlah berkas secara terpisah”*



## File Management System (System File / Sistem Berkas) :

- Pada waktu yang lalu aplikasi database dibangun diatas sistem file
- Kekurangan penggunaan sistem file sebagai penyimpan data:
  - Redundansi / kerangkapan data dan inconsistency
    - Format file yang tidak seragam, kerangkapan data di file-file yang berbeda
  - Sulit dalam mengakses data
    - Perlu program baru untuk mengakses data baru
  - Pengisolasian data — banyak file dengan format yang berbeda
  - Masalah integrasi (keterpaduan)
    - Pengendalian terpadu menjadi bagian dari program
    - Sulit untuk menambah elemen pengendali atau mengubah yang sudah ada

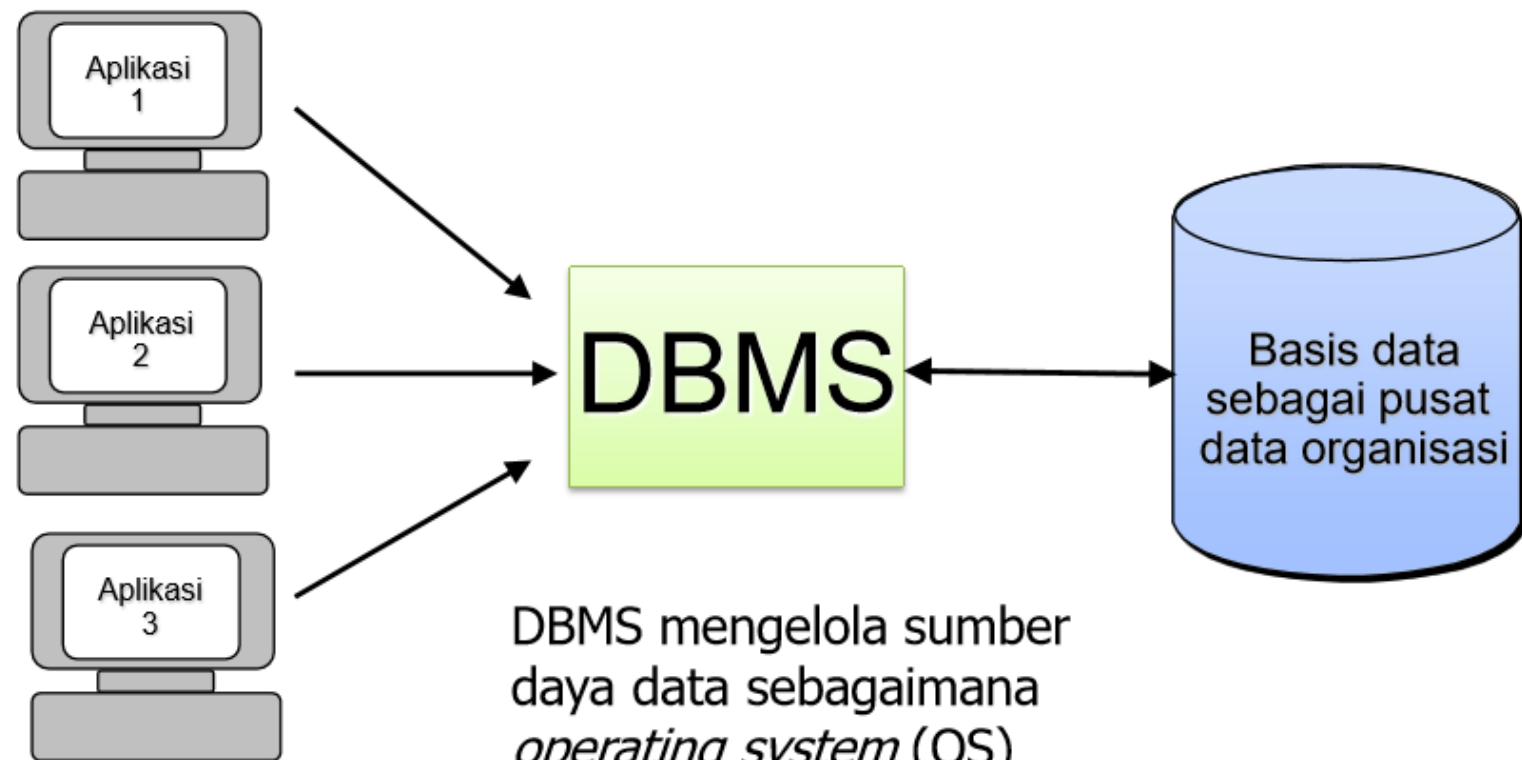
## File Management System (System File / Sistem Berkas) :

- Pengubahan atomik
  - Kesalahan mungkin mengakibatkan database dalam keadaan yang tidak konsisten dengan data yang baru yang dihasilkan
  - Mis. Pengiriman uang dari satu rekening ke rekening yang lain harus terjadi secara lengkap atau tidak sama sekali
- Kesulitan akses secara bersama oleh banyak user
  - Akses secara bersama untuk meningkatkan kinerja
  - Akses bersama akan mengakibatkan ketidak konsistensi  
Mis. Dua orang membaca dan megubah data saldo pada saat yang sama
- Masalah keamanan

Sistem Database mampu mengatasi masalah tersebut diatas

## Pendekatan Basis Data :

# Database Management System (DBMS)



DBMS mengelola sumber daya data sebagaimana *operating system* (OS) mengelola sumber daya perangkat keras

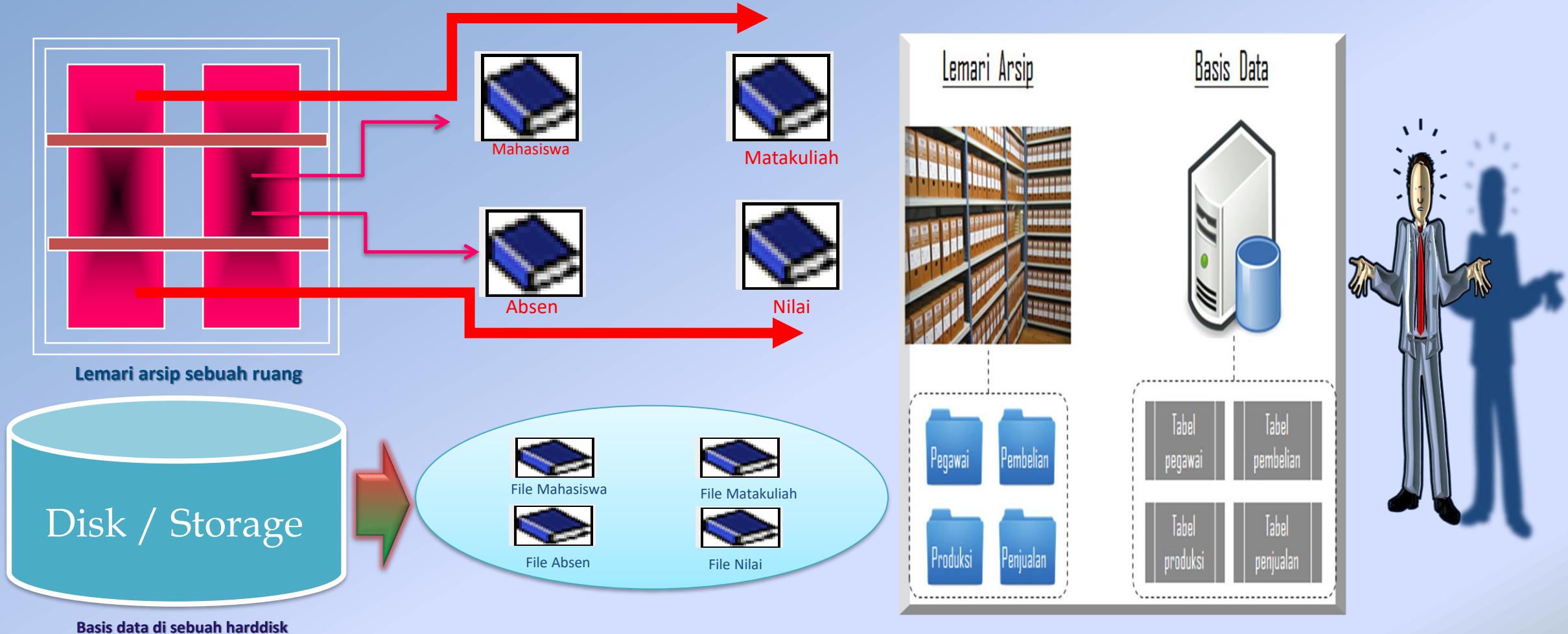
# Database : Rancangan Database Yg Baik (?)

Perancangan Database yg Baik & Benar harus memiliki kriteria sebagai berikut :

- **Anomaly** : Mencegah Concurrent Acces Anomaly (*Insert, Update, Delete*)
- **Redudancy** : Meminimalkan Data Redudancy (*Duplikasi & Inkonsistensi*)
- **Isolation** : Mempertimbangkan Data Isolation (*Overlap Transaction-Multi User*)
- **Security** : Mempertimbangkan Keamanan Data (*Pengelolaan user oleh DBA*).

**Dibutuhkan Normalisasi Database (Terutama utk OLTP) !**

# Database : Sistem Pengarsipan (?)

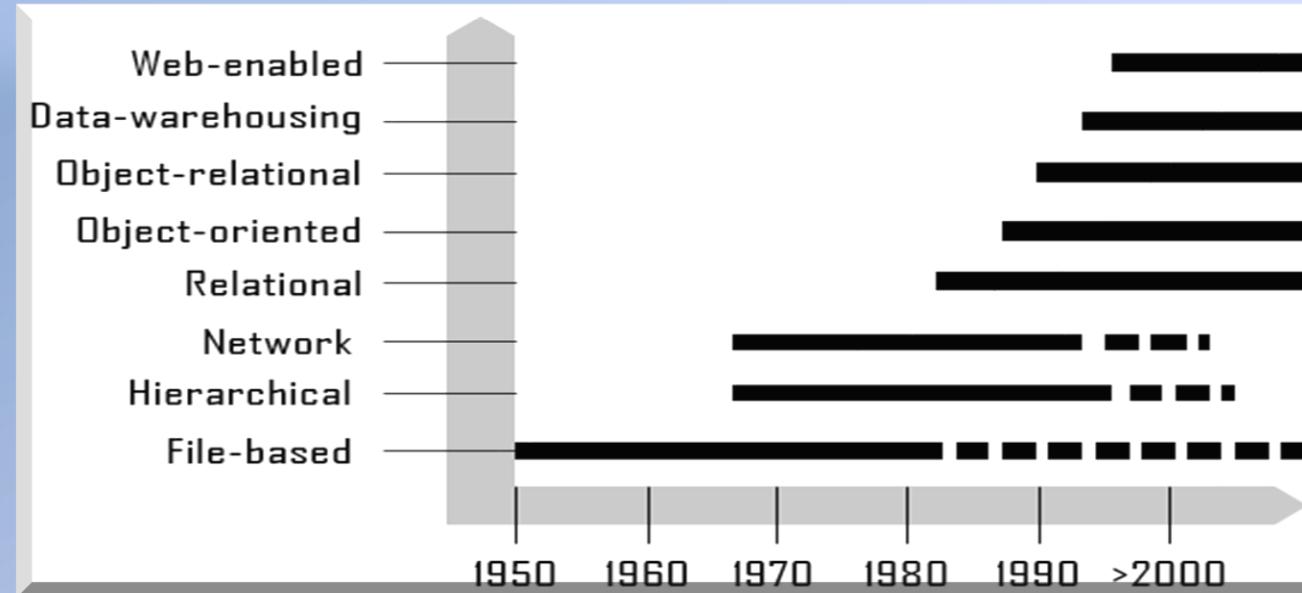


Gambar Ilustrasi Pengarsipan

# Database : Evolusi Teknologi Basis Data (?)

Dipengaruhi oleh Evolusi Fungsi Teknologi Komputer (*File Base System* → *Database Management System*) :

- Evolusi *Data Processing*,
- Evolusi *Information Management*,



Gambar : Evolusi Teknologi Sistem Basis Data

# Database Model : Konsep Pengelolaan Database (?)

**Data model** : adalah sekumpulan konsep yang digunakan untuk menjelaskan data

Merupakan jenis **model data** yg menentukan : 1). struktur logis dari suatu basis data,  
2). Dasar pengelolaan (data disimpan, diorganisir, dimanipulasi).

Berkaitan dengan **konsep database** : integrasi secara logika dari record-record yang bersumber dari berbagai lokasi fisik/storage. Konsep database yang baik harus didukung kamus data (***data dictionary***) , dimana setiap obyek data yg disimpan dalam DBMS harus didefinisikan dan diskripsikan secara jelas, misalnya kumpulan kode , singkatan, penjelasan, dll.

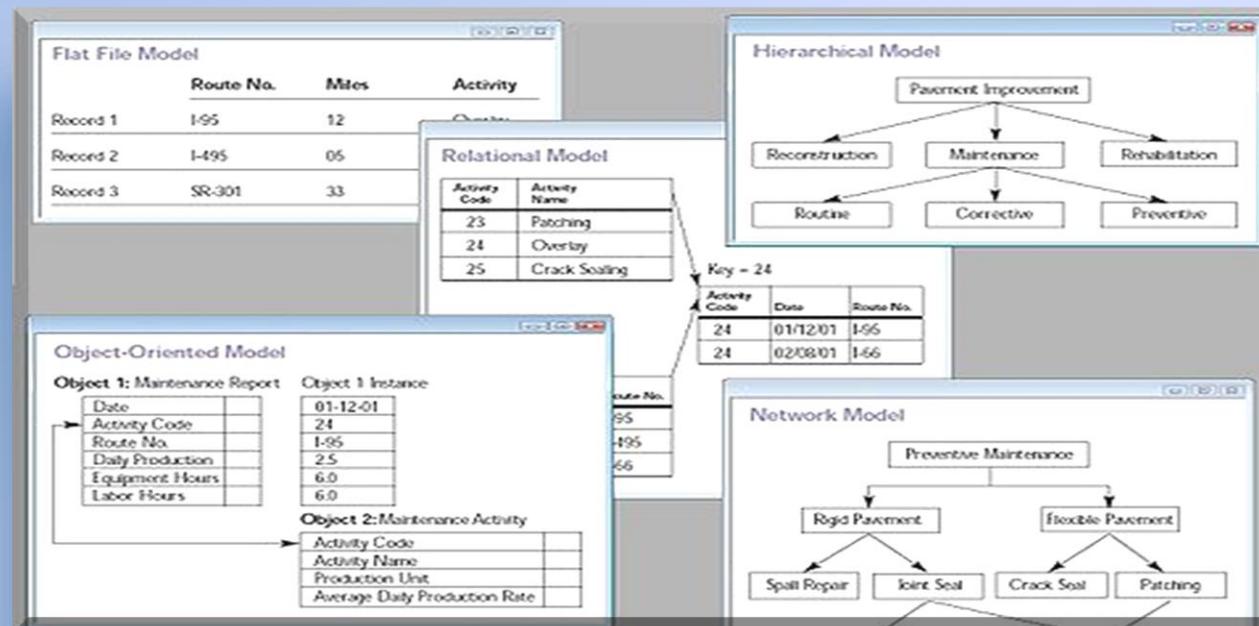
**Tujuan konsep database** : Menghilangkan data redundancy dan menciptakan kemandirian data ( data independence).

Dalam **pengelolaan** (mengolah) database ada beberapa macam konsep atau model yang bisa digunakan antara lain adalah :

- Flat File Model (*SpreadSheet*).
- Model Hierarchical (*Dikembangkan IBM 1960-an*)
- Model Network (*Database Management System, Dikembangkan CODASYL 1969-an*)
- Model R-DBMS (*Relational-Database Management System , Dikembangkan IBM 1970-an*)
- Model Berbasis Obyek (*Object Oriented Model*)

# Database Model : Konsep Pengelolaan Database (Con't)

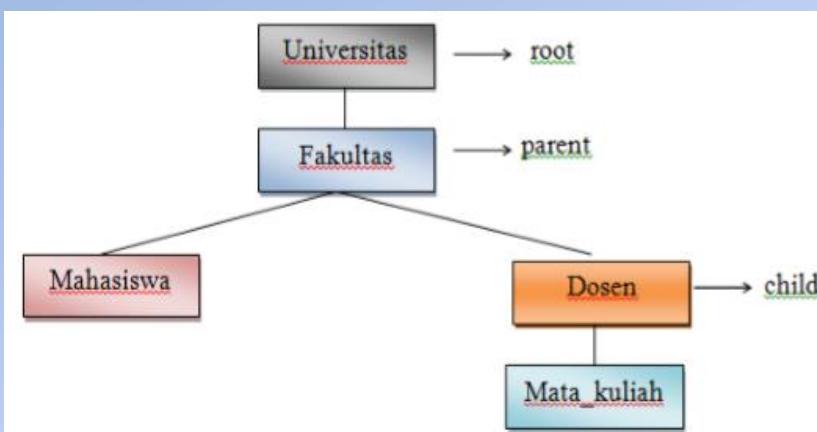
Diantara konsep model tersebut, konsep yang paling popular dan banyak digunakan adalah konsep *R-DBMS*. Konsep ini merupakan model yang paling popular diantara 3 model diatas, (*sudah menerapkan primary key yg disimpan ditabel*) sehingga data lebih mudah diakses dan diatur oleh system. Selain itu konsep ini juga sederhana sehingga mudah dipahami yang terdiri dari baris dan kolom (*tuple dan attribute / raw dan column / record dan field*).



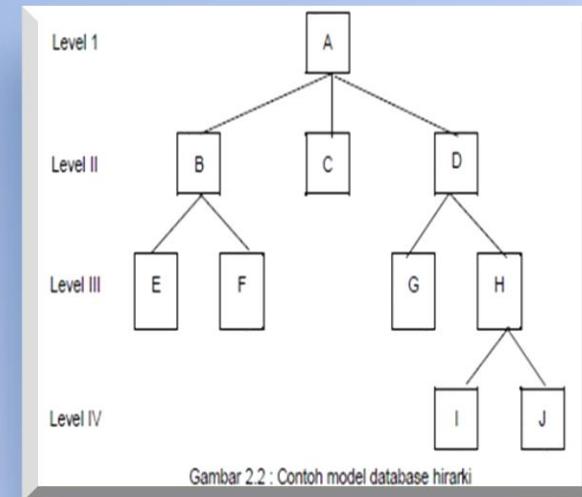
# Model Database : *Hierarchical Database*

Dalam model ini data disusun menurut struktur pohon. Puncak dari herarki disebut dengan **root** sedangkan entitas atau interface di bawahnya dikenal sebagai induk (*parent*).

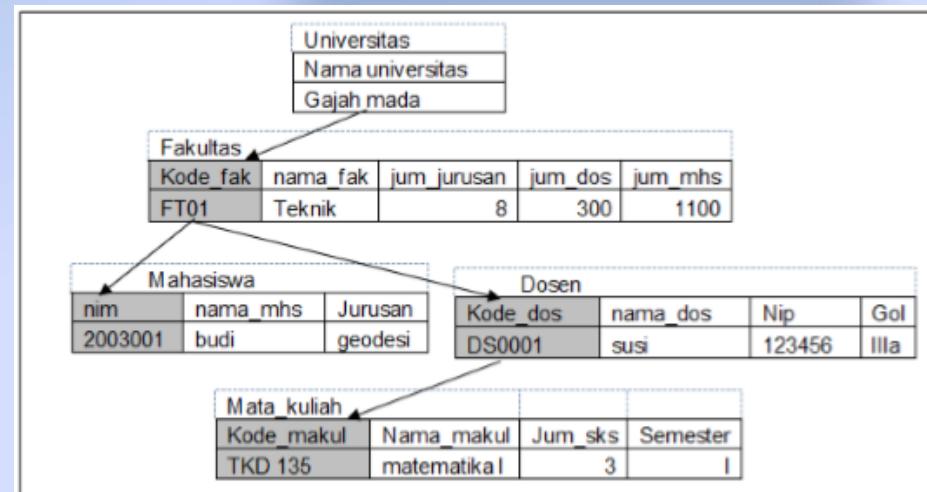
Entitas induk mempunyai beberapa sub entitas yang disebut anak (**child**). Entitas dalam **model hirarki** dilambangkan dengan empat persegi panjang. Sedangkan relasi atau hubungan dengan entitas lain dinotasikan dengan garis.



Model Herarki sistem perkuliahan (level konseptual)



Gambar 2.2 : Contoh model database hirarki

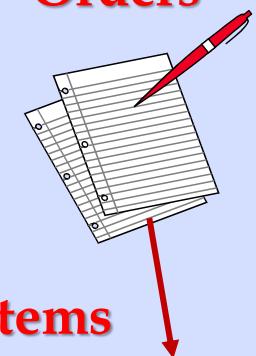


Struktur pengkodean record data (*level fisik*) untuk setiap entitas beserta hubungan antar data. Susunan hirarkhi ditunjukkan dengan tanda anak panah pada median data (*field*) yang digunakan sebagai kunci data (*primary key*, *daerah diarsir*).

**Customers**



**Orders**



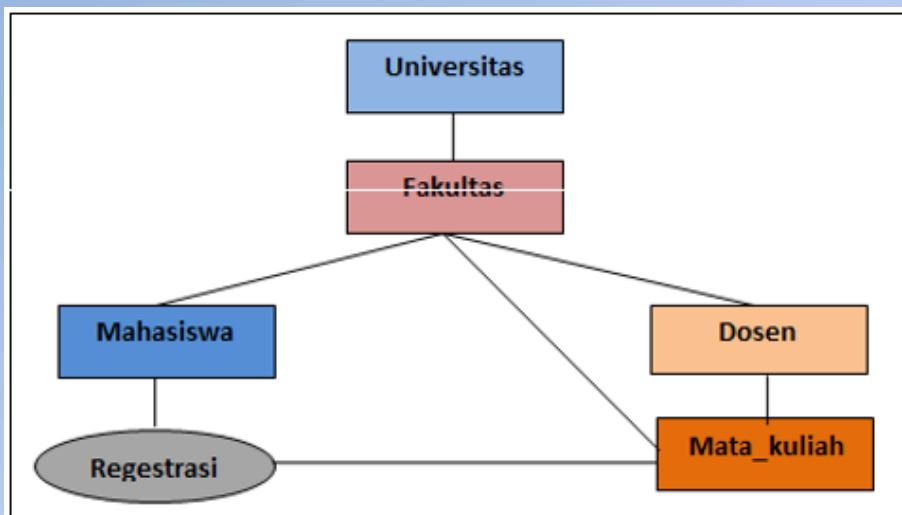
**Items**

Item	Description	Quantity
998	Dog Food	12
764	Cat Food	11

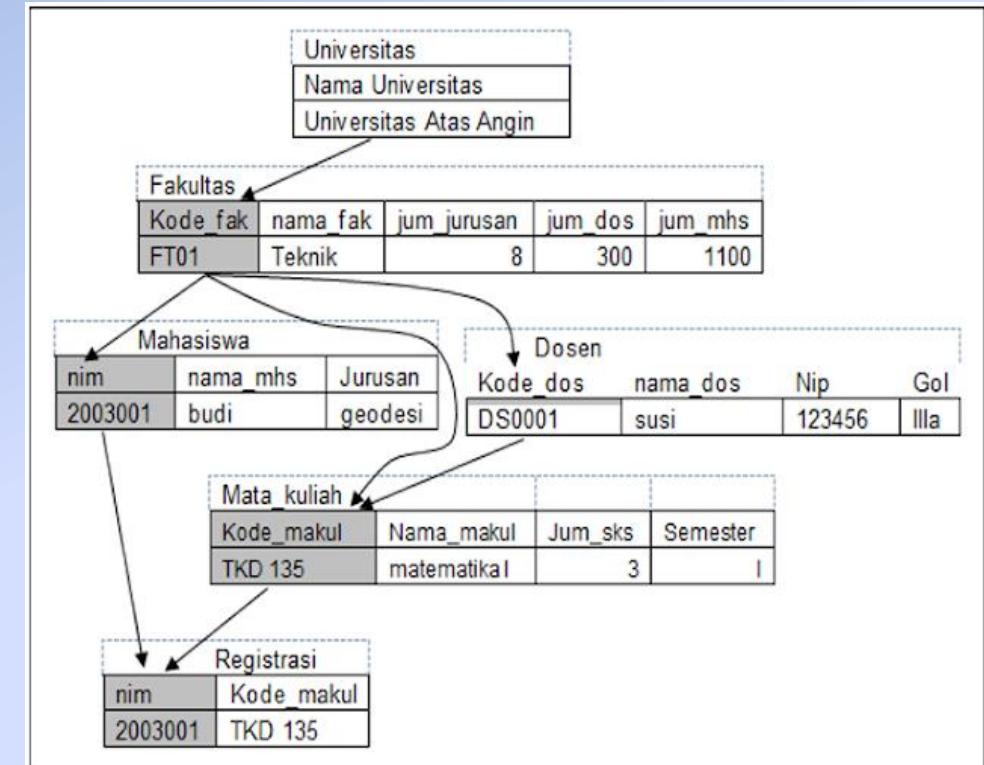
Untuk mengambil data, Anda harus mulai dari atas (**Customer**). Saat Anda mengambil pelanggan, Anda mengambil semua data bersarang.

# Model Database : *Network Database*

Dalam model jaringan entitas induk maupun anak bisa lebih dari dua. Model ini merupakan pengembangan **model hirarki**. Relasi antara entitas dalam network model adalah satu ke satu (*one to one*) atau satu ke banyak (*one to many*)



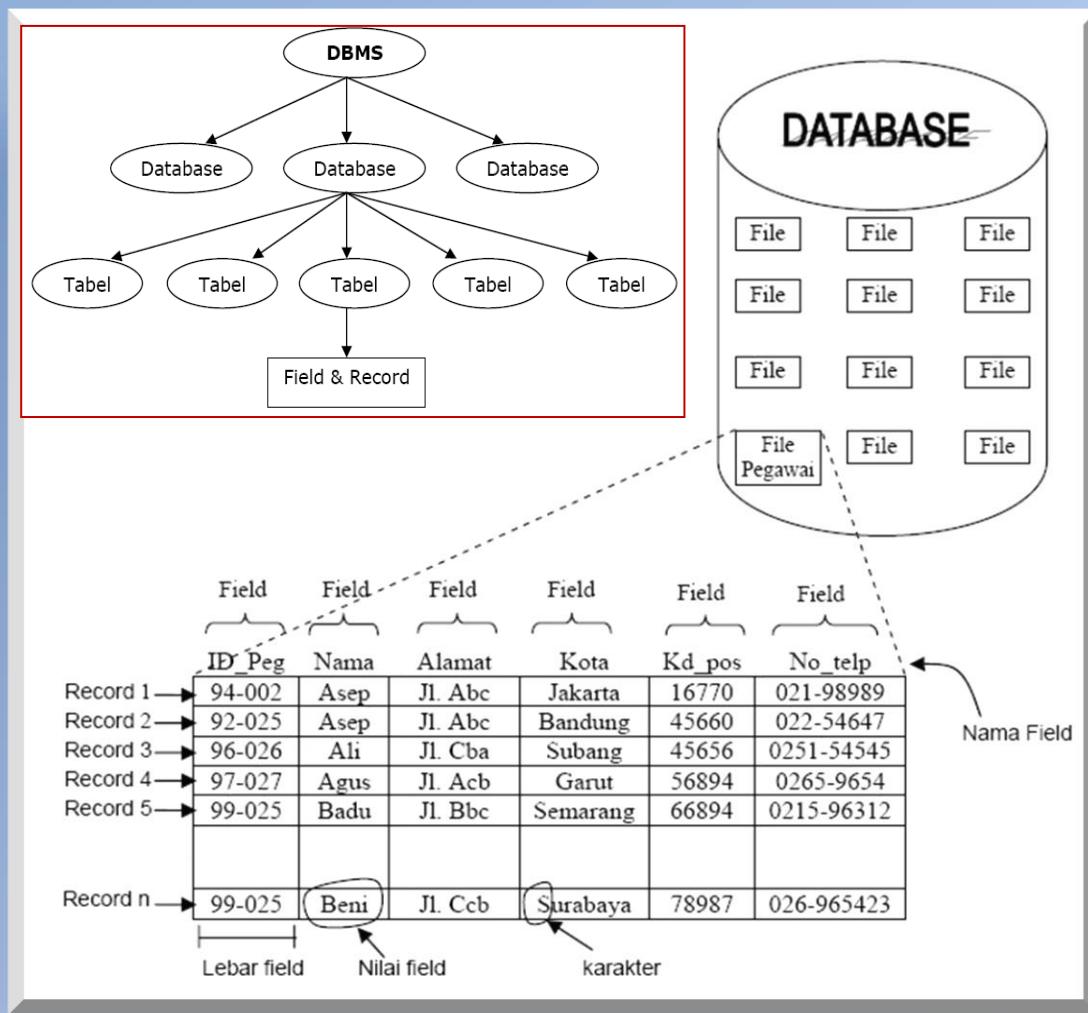
Dalam network data model tidak diperbolehkan terdapat relasi banyak ke banyak (*many to many*). Untuk membuat relasi many to many dalam network model dibutuhkan **entitas perantara** yang disebut sebagai rekaman silang (*intersection record*). Dari gambar diatas entitas registrasi adalah merupakan entitas perantara antara entitas mahasiswa dengan entitas mata kuliah.



Model jaringan merupakan pengembangan model hirarki. Dalam model ini entitas induk maupun anak dapat memiliki lebih dari dua entitas. Hubungan atau relasi antara entitas dalam network model adalah satu ke satu (*one to one*) atau satu ke banyak (*one to many*). Ciri khas model ini adalah terdapatnya entitas perantara yang disebut sebagai rekaman silang (*intersection record*). Entitas perantara berfungsi untuk relasi many to many.

# Model Database : *Relational Database*

Basis Data akan disebar / dipilah ke dalam tabel dua dimensi



**Pegawai** (ID\_Peg, Nama, Alamat, Kota, ...)

**Departemen** (Dept\_ID, Nama\_Dept, Kota, ...)

**Bekerja** (ID\_Peg, Dept\_ID, Thn\_Kerja, ...)

**Gaji** (ID\_Peg, Gol\_ID, Tunj\_Kel, ...)

**Golongan** (Gol\_ID, Diskripsi, GajiPokok, ...)

customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

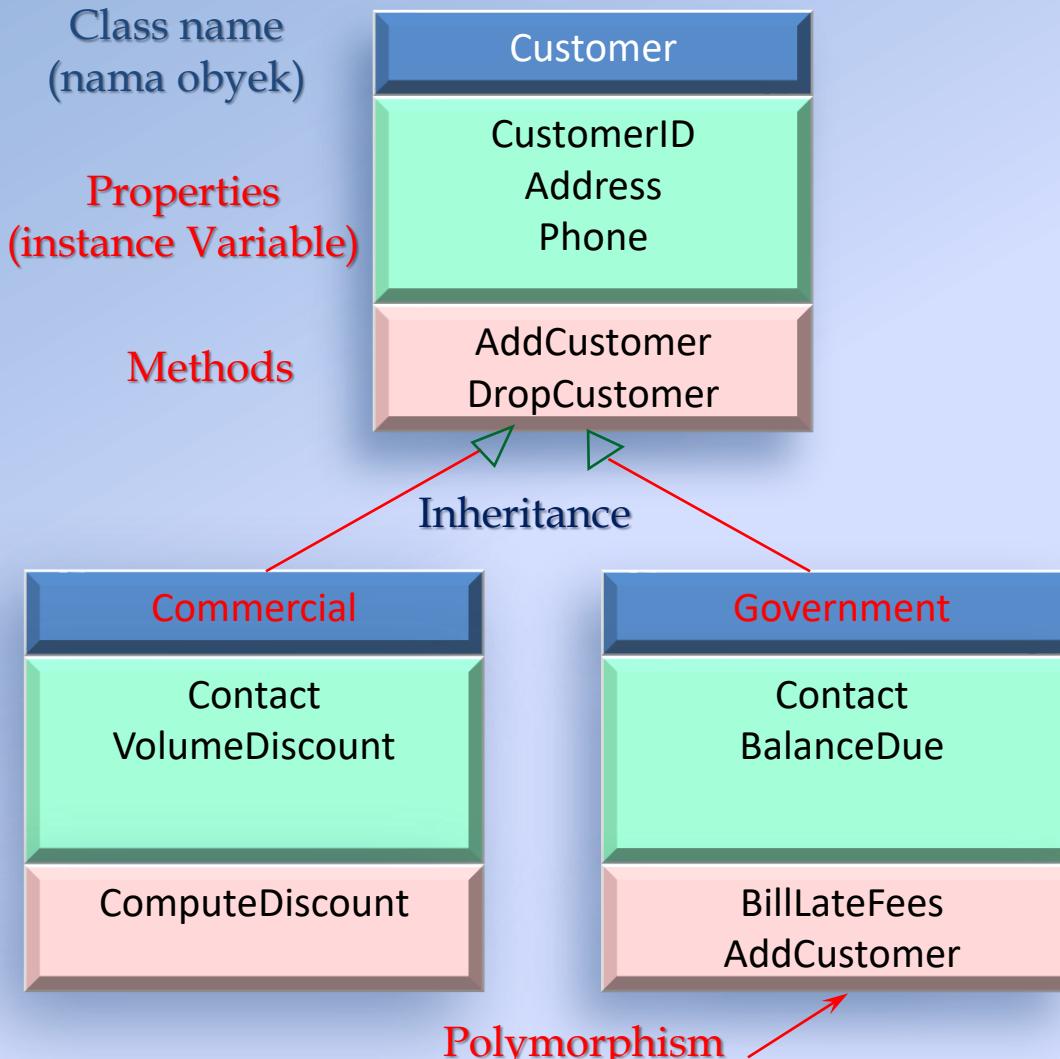
account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

# Database Models : Object Data Model



class dapat memiliki banyak “**bentuk**” method yang berbeda-beda meskipun namanya sama.

# Database : Jenis File Pada DBMS (?)

Jenis-jenis file yang digunakan dalam DBMS dibedakan menjadi :

## ● **File Induk (*master File*)**

- file induk acuan (*reference master file*) : file induk yang recordnya relatif statis, jarang berubah nilainya. Misalnya file Mahasiswa, file Prodi, file mata kuliah.
- file induk dinamik (*dynamic master file*): file induk yang nilai dari record-recordnya sering berubah atau sering dimutakhirkan (*update*) sebagai hasil dari suatu transaksi. Misalnya file induk data barang, yang setiap saat harus di *up-date* bila terjadi transaksi (mandiri).

- **MASTER FILE**

- Adalah:

- Berisi data statis
    - Data tentang satu sisi dari organisasi
    - Berisi data historis
    - Isinya relatif permanen

**PELANGGAN  
PEGAWAI  
MAHASISWA**

**PERSEDIAAN BARANG  
NILAI  
MATAKULIAH**

# Database : Jenis File Pada DBMS (?)

Jenis-jenis file yang digunakan dalam DBMS dibedakan menjadi :

- **File Transaksi (*transaction file*)**

file ini bisa disebut *file input*; digunakan untuk merekam data hasil dari transaksi yang terjadi. Misalnya file penjualan yang berisi data hasil transaksi penjualan.

- **File Laporan (*Report file*)**

File ini bisa disebut *output file*, yaitu file yang berisi informasi yang akan ditampilkan.

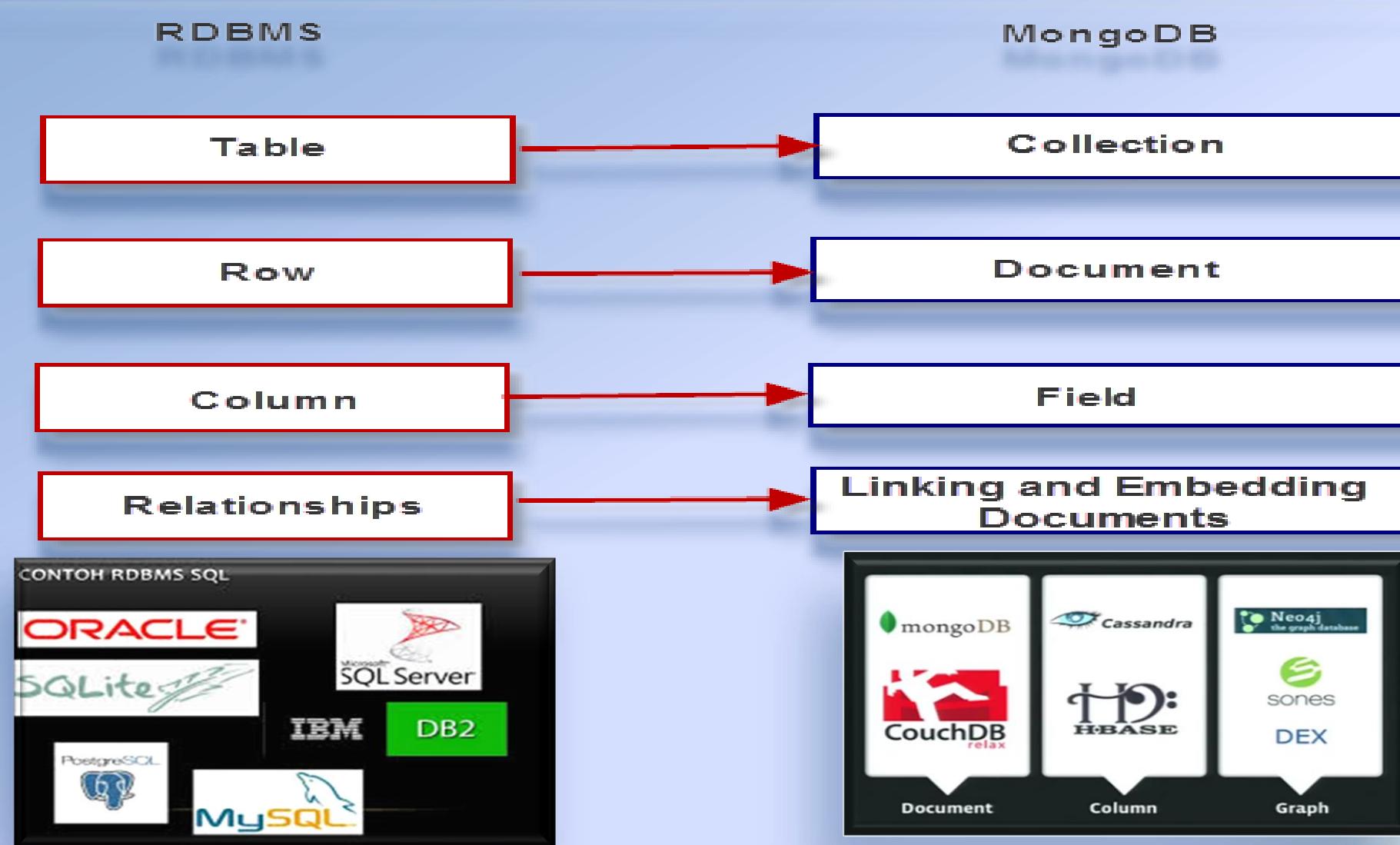
- **File Sejarah (*history file*)**

File ini bisa disebut file arsip (archival file), merupakan file yang berisi data masa lalu yang sudah tidak aktif lagi, tetapi masih disimpan sebagai arsip.

- **File Pelindung (*backup file*)**

File ini merupakan salinan dari file-file yang masih aktif di dalam database pada suatu saat tertentu. File ini digunakan sebagai pelindung atau cadangan bila file database yang aktif mengalami kerusakan atau hilang.

# Database : SQL VS NoSQL(big Data/Cloud)



# Persamaan & Perbedaan : DBMS & RDMBS

- **Persamaan :** DBMS & RDBMS adl. S/W yg memiliki fungsi yang sama, yaitu : menyimpan & menyajikan data (informasi ) dari suatu database (dlm. bentuk tabel).
- **Perbedaan :**  
RDBMS adalah arsitektur **database** yang tabel-tabelnya mempunyai hubungan atau relationship satu sama lain. Hubungan disini menggunakan key pada masing-masing tabel. Sedangkan kebalikannya, **DBMS** tidak harus membutuhkan hubungan antar tabel di dalamnya.
- RDBMS adalah varian dari DBMS yang dirancang untuk menghilangkan inefisiensi DBMS.

	<b>DBMS</b>	<b>RDMBS</b>
Contoh	Dbase, Microsoft Access, Pangkalan LibreOffice, FoxPro.	SQL server, Oracle, mysql, MariaDB, SQLite.

# RDBMS : Contoh Memodelkan Dunia Nyata (?)

- Basis data memodelkan “dunia nyata” yang berkaitan dengan :

- ✓ **Entities** : sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lainnya (individu : manusia, tempat, obyek, kejadian, konsep) Contoh: Mahasiswa, Progdi, Matakuliah, dll.
- ✓ **Relationships** (contoh: Berlian L. mhs Prodi-PJJ Ilkom Kuliah mengambil matakuliah Konsep Database dpt Nilai Tugas 70, Kuis, 75, UTS, 75, UAS 85)

Tabel Mahasiswa

Nim	Nm_Mhs	Kelamin	Kd_Prodi
A1-001	Berliana Lina	Wanita	A11.261
A1-002	Cycal	Pria	A11.261
A-1003	Tiara	Wanita	A11.262
A-1004	Jenar	Pria	A11.262

Tabel Progdi

Kd_Prodi	Nm_Progdi
A11.261	PJJ-Ilmu Komputer
A11.262	Sistem Komputer
A11.260	Mnj. Informatika
A11.263	Teknik Informatika

Tabel Kuliah

Nim	Kd_MK	Nilai_Tugas	Nilai_Kuis	Nilai_UTS	Nilai_UAS
A1-001	PJKI52001	70	75	75	90
A1-001	PJKI53002	75	80	90	85
A1-002	PJKI52001	80	65	80	80
A1-002	PJKI52002	75	70	75	90
A1-003	PJKI53003	90	80	70	85
A1-004	PJKI53003	100	75	90	90

Tabel Matakuliah

Kd_MK	Nm_MK
PJKI52001	Konsep Database
PJKI53002	Sistem Operasi
PJKI53003	Sistem Distribusi

# RDBMS : Memodelkan Dunia Nyata (cont'd)

Tabel barang direlasikan dg tabel distributor melalui Kode\_Dist, Tabel Barang dg Tabel Transaksi\_Jual melalui Kd\_Brg dan Tabel Pelanggan dg Tabel Transaksi\_Jual melalui Kd\_Plgn.

Memungkinkan pencarian informasi yg melibatkan seluruh file yg berelasi (?)

Tabel Barang

Kd_Brg	Nm_Brg	Harga_Beli	Kd_Pemasok
100	Pisau Roti	14500	10
110	Roti Tawar	24600	10
120	Blueband Tr	44500	20
130	Selai Strobery	24500	30

Tabel Distributor

Kd_Dist	Nm_Pemasok
10	Unilaper
20	Prima Sari
30	Bintang Lina
40	Sekawan Purwa

Tabel Transaksi\_Jual

Kd_Brg	Kd_Plgn	No_Fact	Tgl_Trans	Jumlah	Harga_Jual
100	220	111	29/09/2019	40	15000
110	220	112	30/09/2019	15	25000
120	221	113	31/09/2019	25	45000
130	221	114	31/09/2019	15	25000
100	222	113	30/09/2019	10	15000
110	222	114	29/09/2019	35	25000

Tabel Pelanggan

Kd_Plgn	Nm_Plgn
220	Brindys T
221	Cycal H
222	Jenar J

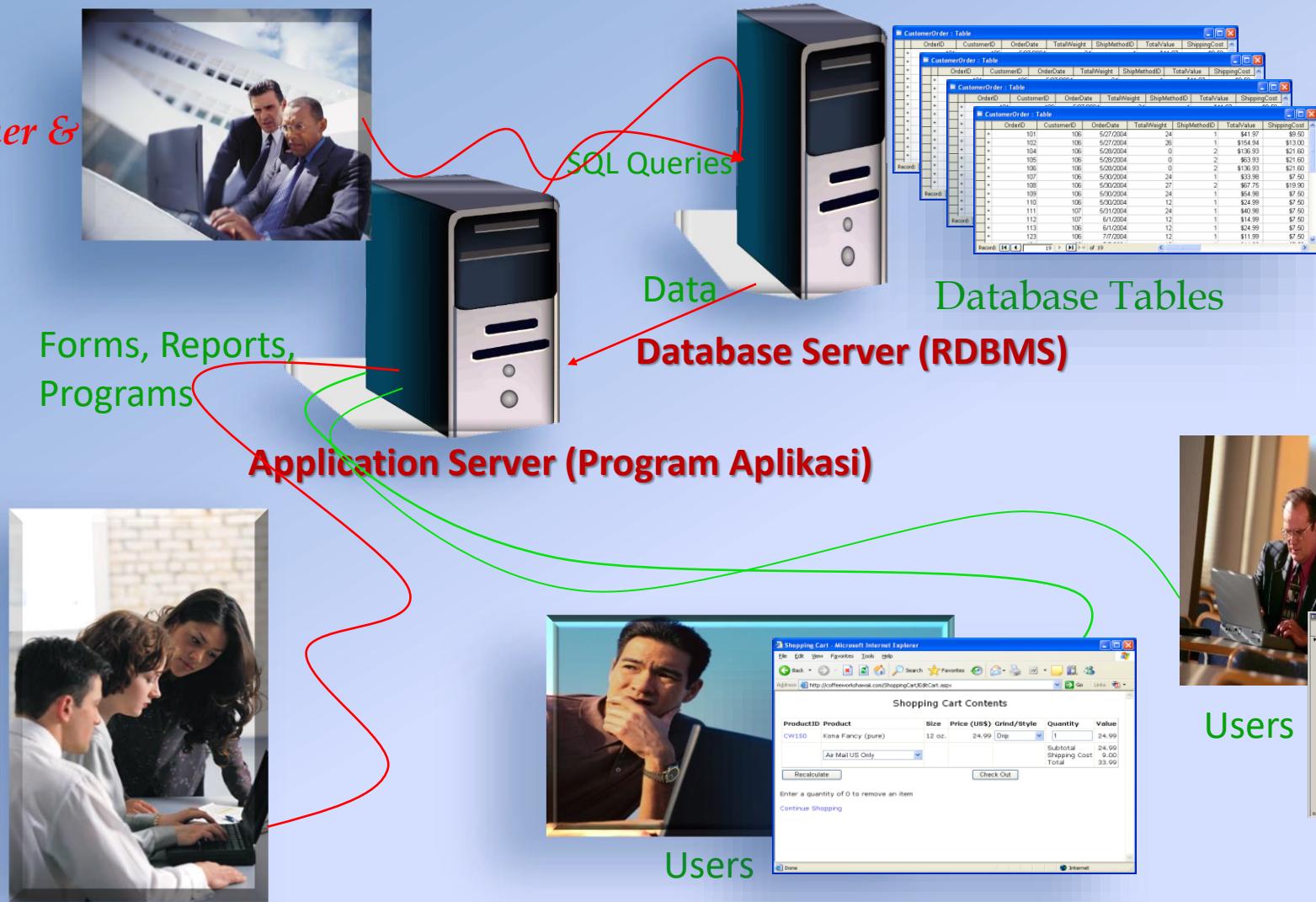
# RDBMS : Fungsi (?)

- **Secara umum :** relasional database adalah jenis database dan RDBMS adalah software (S/W) yang mengelolanya. Sedangkan (**RDBMS**) sendiri adalah sebuah system agar data yang ada di database tersebut tetap konsisten dan terjaga.
- **Secra khusus fungsi RDBMS adalah :**
  - Membuat banyak tabel di dalam database
  - Membaca data yang ada di database
  - Memperbaiki struktur susunan database
  - Menghapus struktur yang sudah tidak digunakan lagi di database

Keempat fungsi dasar tadi sebenarnya sama dengan fungsi dasar **CRUD** istilah yang sudah tidak asing lagi di telinga kita yaitu : *Create, Read, Update, Delete.*

# Database : Data Aplikasi Disajikan Dari RDBMS (?)

Database Desainer & Administrators



Developers/Programmer

Application Forms

Application Forms

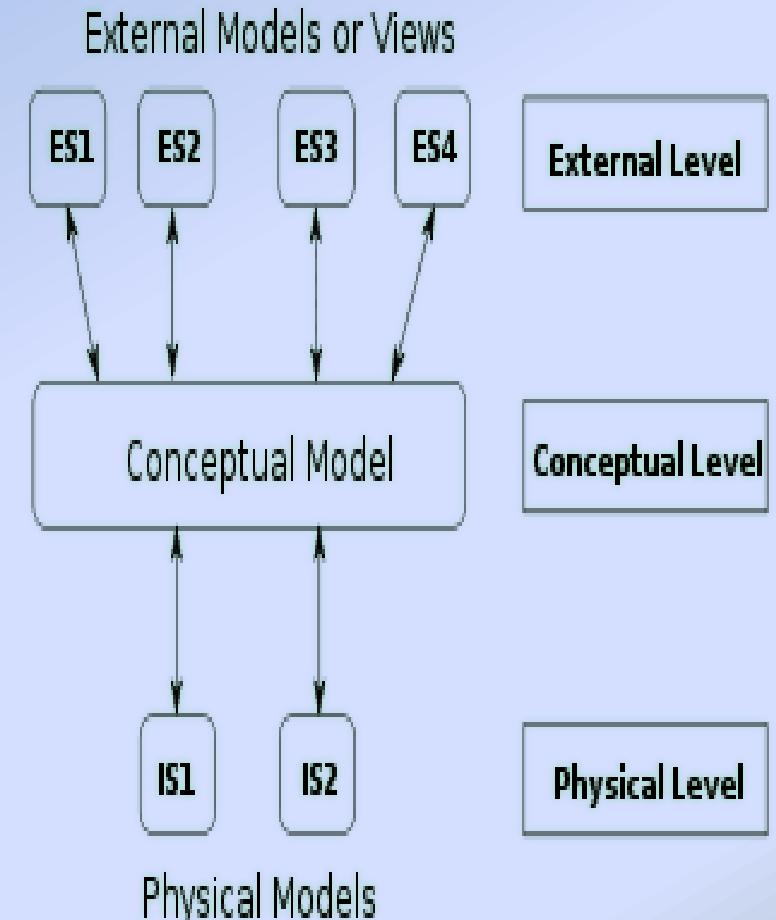
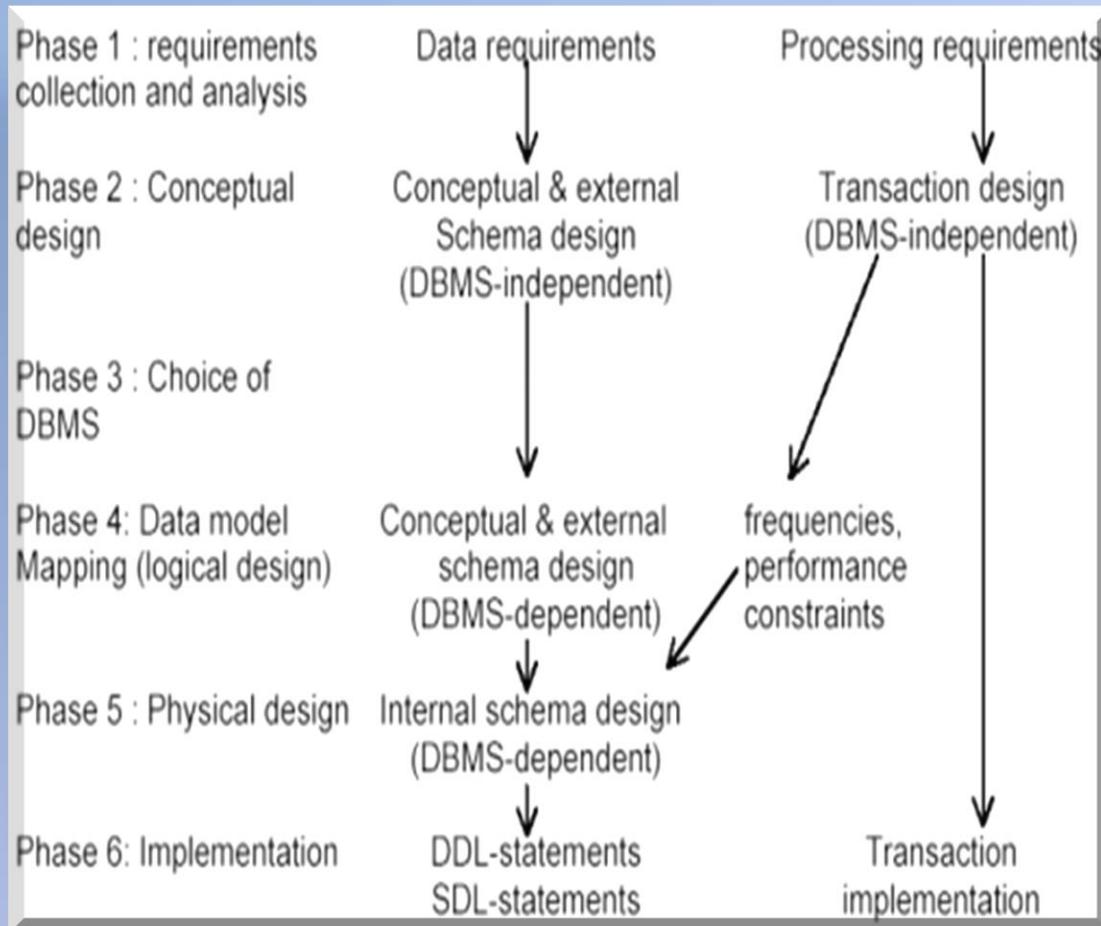
# Database : Langkah Rancangan Database (?)

Meliputi perancangan pemilihan RDBMS baik secara *Conceptual, Logical* dan *Physical*. Langkah-langkah dalam mendesign database (database design). Ada 3 langkah utama dalam design database yaitu :

1. **Conceptual Model** : Adalah mendefenisikan data-data yang diperlukan (organisasi). Pada langkah pertama ini perlu diperhatikan adalah data apa yang dibutuhkan sebagai output, baik melalui screen (layar) atau printer, yang datanya harus disimpan dalam file database. (Analisis : Mendiskripsikan semantik suatu domain, Identifikasi & Analisa kebutuhan (data / informasi ) & aturan bisnis.) → Conceptual Schema Desain, ER-Diagram.
2. **Logical Database Design** : Adalah menentukan data yang akan dikelompokan dalam suatu file database. Hal yang penting dalam pengembangan adalah logika desain database, model relasi dan proses normalisasi yaitu pengelompokan data menjadi satu tabel berdasarkan entity dan relasinya. → Mapping ER-Diagram to Relational-Model/Data Model Mapping.
3. **Physical Database Design** : Adalah pertimbangan kemampuan sistem yang akan dipakai jika diperlukan pembahasan yang sesuai dengan informasi sistem. Pada tahap ini adalah untuk mempertimbangkan, penyisipan dan penghapusan jika tidak terjadi maka tabel-tabel yang telah dinormalisasikan harus digabungkan kembali yang disebut penormalisasian. → Implementasi dan Diskripsi Model Relational Sebagai File Kedalam Lingkungan DBMS Pilihan (Mempertimbangkan : Respone Time, Space Utility, Transaction Throughput).

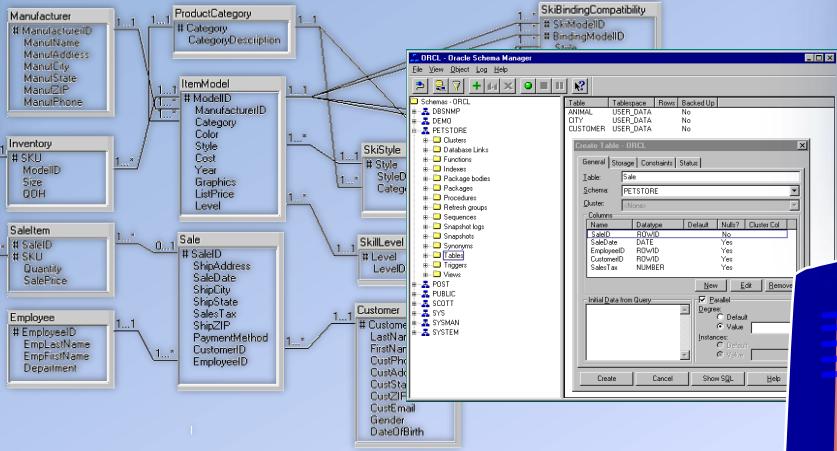
# Database : Fase Perancangan utk Database Besar (?)

Terdiri 2 aktifitas parallel, yaitu aktifitas perancangan dari isi data dan struktur basis data, dan aktifitas perancangan pemrosesan basis data dan aplikasi-aplikasi perangkat lunak

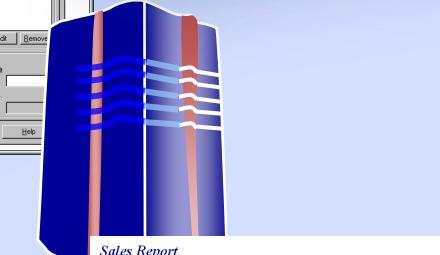


# Database : Desain Aplikasi RDBMS (Cont'd)

1. Identifikasi & analisa kebutuhan data serta aturan bisnis.



2. Tentukan tabel dan relasi.



3. Buat form Inputan dan laporan.

A screenshot of a 'Sale' input form. It shows a table with columns: SaleID, SaleDate, Customer, Employee, SaleTime, SaleTax, and SalePrice. Below the table is a grid showing items sold, including 'Dog Kennel Small' and 'Leash'. At the bottom, there are summary totals for 'Merchandise Total' (\$183.38) and 'Sales Tax' (\$14.62).

4. Kombinasikan sebagai aplikasi untuk pengguna.

# File Systems v.s. RDBMS

- Utk. memahami kebutuhan akan RDBMS, perhatikan suatu skenario berikut :

- Perusahaan mempunyai basis data berukuran besar, katakanlah sebesar 800 GB, untuk menyimpan dan memelihara data karyawan, departemen, produk, customer, keuangan, penjualan, dll.
- Data tsb harus dapat diakses secara serentak (**concurrent**) oleh beberapa karyawan/user,
- Pertanyaan (**queries**) mengenai data hrs dapat dijawab secara cepat
- Perubahan-perubahan terhadap data (**update**) oleh sejumlah pengguna yang berbeda hrs dapat dilakukan secara konsisten.
- Akses ke bagian-bagian tertentu dari data (misalnya, data gaji) hrs dibatasi (**restricted**).

Untuk Mengelola Data Perusahaan Tersebut :  
*Spreadsheet/Excel ? Database ?*

## **File Systems v.s. RDBMS (Cont'd)**

**Data tsb dapat saja disimpan dalam file systems suatu sistem operasi. Namun cara ini dapat memiliki banyak kelemahan seperti berikut :**

- Aplikasi hrs secara eksplisit memilah data yang besar antara main memory dan secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Hrs menulis program-program yang khusus untuk berbagai queries yang berbeda.
- Hrs memproteksi data terhadap terjadinya inkonsistensi akibat akses banyak pengguna secara serentak.
- Hrs menyediakan pemulihan kembali terhadap terjadinya "crash" dari sistem.
- Persoalan sekuritas dan pengendalian akses yang kurang fleksibel, karena sistem operasi biasanya hanya menyediakan mekanisme "password" untuk kebutuhan sekuritas sistem.

# Mengapa Menggunakan RDBMS (?)



Beberapa kelebihan atau keuntungan RDBMS dalam pengelolaan data

- Data independence
- Efficient data access
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes
- Reduced application development time



# Beberapa Alasan Utk **TIDAK Menggunakan RDBMS !**

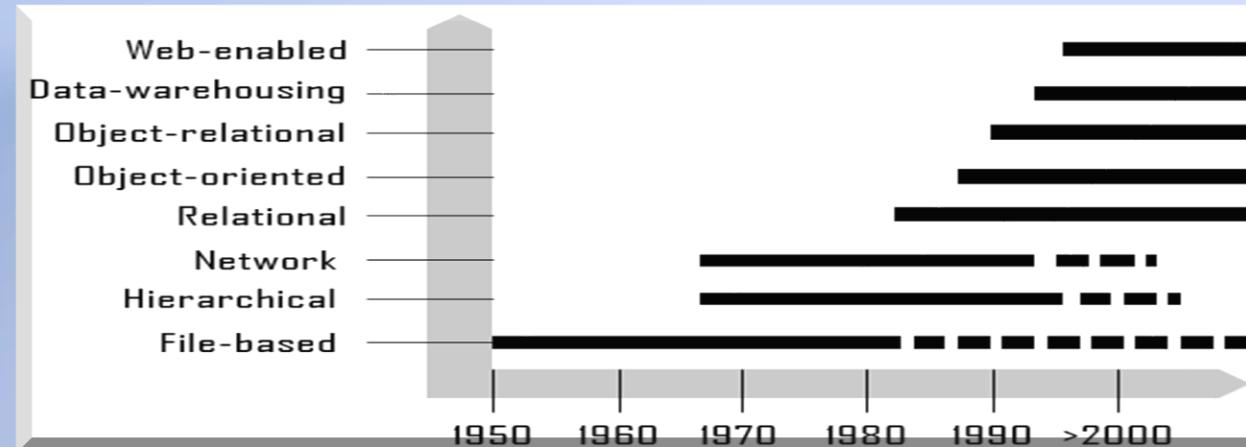
- **Kinerja RDBMS mungkin tidak cukup utk beberapa aplikasi khusus, misalnya:**
  - Aplikasi dengan batasan “real-time” yang ketat.
  - Aplikasi yang memerlukan operasi-operasi kritis yang melibatkan program-program yang harus ditulis secara khusus.
- **Aplikasi mungkin memerlukan manipulasi data dengan cara yang tidak didukung oleh “query language” dalam suatu RDBMS**
  - Sebagai contoh, relationsl DBMS biasanya tidak menyediakan fasilitas untuk melakukan analisis data teks secara fleksibel.
- **Aplikasi tidak memerlukan “added benefits” yang ditawarkan oleh RDBMS**

# Mengapa Hrs Belajar Basis Data (?)

- Pergeseran paradigma dari “*komputasi*” to “*informasi*”
- Peningkatan keaneragaman dan volume data
  - Digital libraries, interactive video, dlsb .... memerlukan ledakan RDBMS
- RDBMS meliputi sebagian besar dari “ilmu komputer”
  - Sistem operasi, bahasa pemrograman, multimedia, kecerdasan buatan, dlsb.

Dipengaruhi oleh Evolusi Fungsi Teknologi Komputer (*File Base System* → *Database Management System*) :

- Evolusi *Data Processing*,
- Evolusi *Information Management*,



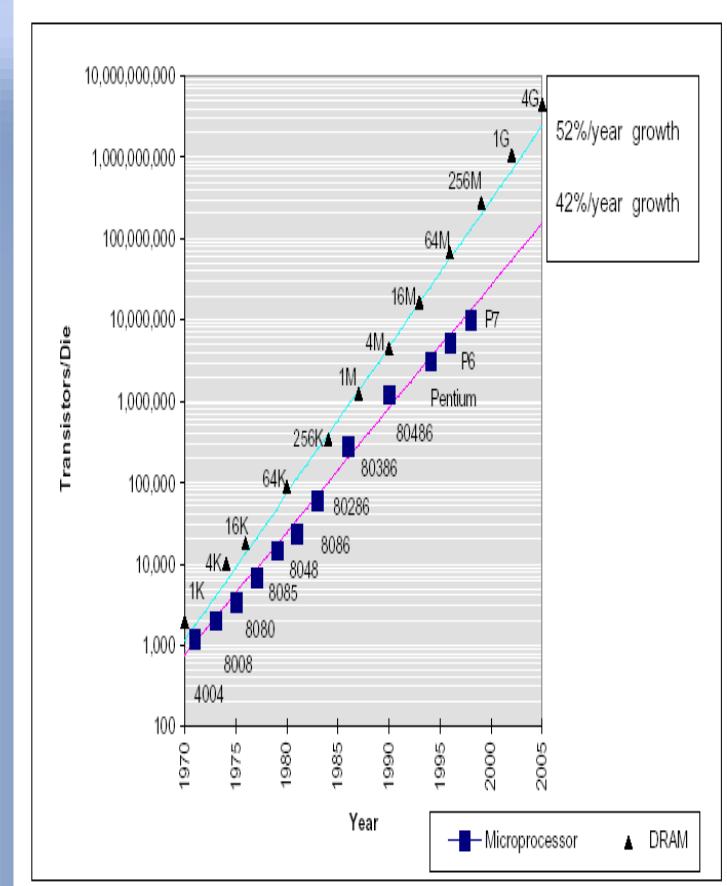
Gambar : Evolusi Teknologi Sistem Basis Data



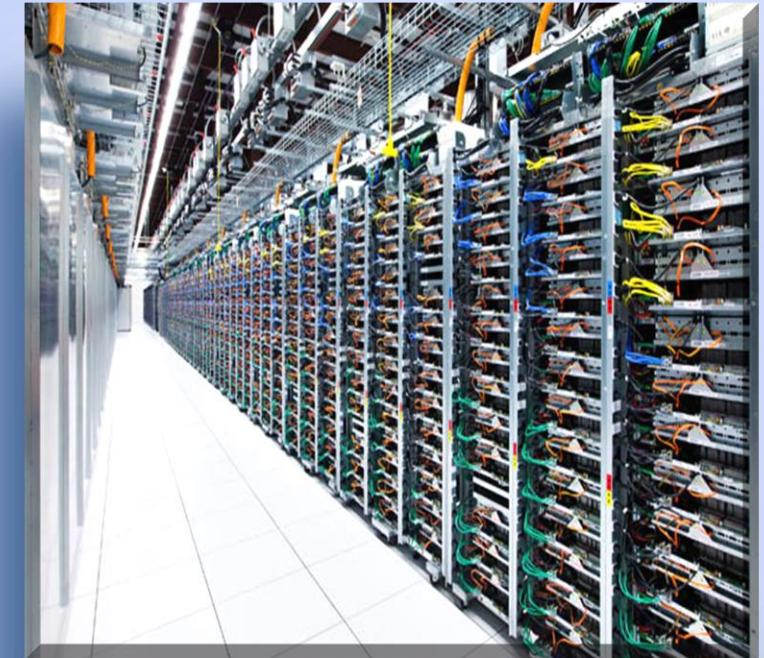
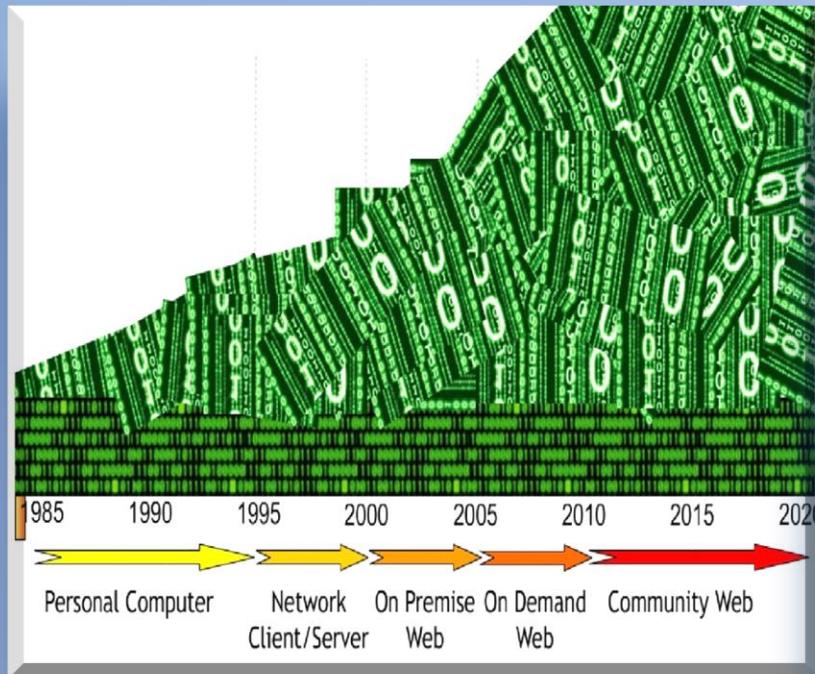
# Perkembangan :

## RDBMS, Storage, Microprocessor, DRAM

No.	Jenis Produk	Vendor	Kapasitas
1.	ORACLE	Oracle	8 Exabyte's
2.	DB2	IBM	4 Exabyte's
3.	SYBASE	Power Soft	2 Exabyte's
4.	INFORMIX	Informix	2 Exabyte's
5.	MYSQL	SQL Maestro Group	1 Exabyte's
6.	MS ACCESS	Microsoft	650 Terabytes
7.	MS SQL Server	Microsoft	4 Exabyte's
8.	Firebird	FirebirdSQL Corp	2 Exabyte's
9.	PostgreSQL	PostgreSQL Corp	2 Exabyte's
10.	Interbase	Interbase Corp	4 Exabyte's
11.	MongoDB	MongoDB, Inc. 2015	180 Exabyte's



# Fakta Pertumbuhan Data : Data Center, Big Data



# Data Models : Model Relational

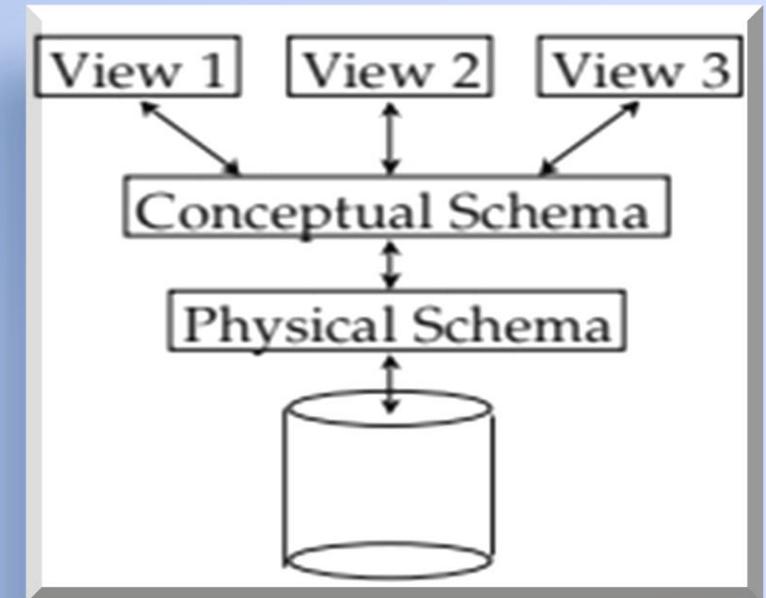
- **Data model** : adalah sekumpulan konsep yang digunakan untuk menjelaskan data
  - **Schema** : adalah deskripsi dari sekumpulan data dengan menggunakan suatu data model tertentu
  - **Relational data model** : adalah model data yang paling banyak digunakan pada saat ini
- 
- Konsep utama: **relasi (relation)**, yang pada dasarnya berupa “tabel” yang terdiri dari sejumlah “baris” dan “kolom”
  - Setiap relasi mempunyai sebuah **skema (schema)**, yang menjelaskan kolom-kolom (***fields***) dari sebuah tabel
  - Contoh, informasi mahasiswa (***students***) di suatu universitas dapat disimpan dalam sebuah relasi dengan skema **Students**(***sid***: string, ***name***: string, ***login***: string, ***age***: integer, ***gpa***: real)

<b><i>sid</i></b>	<b><i>name</i></b>	<b><i>login</i></b>	<b><i>age</i></b>	<b><i>gpa</i></b>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

# Levels of Abstraction

- Terdiri dari BANYAK views, SATU conceptual (logical) schema dan SATU physical schema.

- **Views (External schemas)** menjelaskan bgm pengguna melihat data
- **Conceptual schema** mendefinisikan struktur logikal
- **Physical schema** menjelaskan detil penyimpanan data (misalnya, files, indexes, struktur-struktur penyimpanan fisik, penempatan record dan jalur akses)



struktur-struktur penyimpanan fisik, penempatan record dan jalur akses

\* *Schema didefinisikan menggunakan DDL, sedang queries/modifikasi terhadap data dilakukan dengan menggunakan DML*

# Contoh : Database Akademik

## ● Conceptual Schema :

- Mahasiswa(nim: string, nm\_mhs: string, kelamin: string, kd\_prodi: string)
- Progdi(kd\_prodi: string, nm\_progdi: string)
- Matakuliah(kd\_mk: string, nm\_mk: string, sks: integer)
- Kuliah(nim: string, kd\_mk: string, nilai\_tugas: real, nilai\_kuis:real, nilai\_uts: real, nilai\_uas: real)

## ● Physical Schema :

- Relations disimpan dalam bentuk “unordered files”.
- Index pada kolom pertama dari relasi Mahasiswa.

## ● External Schema (View) :

- Info\_nilai(nim: string, kd\_mk: string, nilai\_uas)

# RDBMS : Contoh Memodelkan Dunia Nyata (?)

- Basis data memodelkan “dunia nyata” yang berkaitan dengan :
  - ✓ *Entities* (contoh: Mahasiswa, Progdi, Matakuliah, dll.)
  - ✓ *Relationships* (contoh: Berlianana L. mhs Prodi-PJJ Ilkom Kuliah mengambil matakuliah Konsep Database dpt Nilai Tugas 70, Kuis, 75, UTS, 75, UAS 85)

Tabel Mahasiswa

Nim	Nm_Mhs	Kelamin	Kd_Prodi
A1-001	Berliana Lina	Wanita	A11.261
A1-002	Cycal	Pria	A11.261
A1-1003	Tiara	Wanita	A11.262
A1-1004	Jenar	Pria	A11.262

Tabel Progdi

Kd_Prodi	Nm_Progdi
A11.261	PJJ-Ilmu Komputer
A11.262	Sistem Komputer
A11.260	Mnj. Informatika
A11.263	Teknik Informatika

Tabel Kuliah

Nim	Kd_MK	Nilai_Tugas	Nilai_Kuis	Nilai_UTS	Nilai_UAS
A1-001	PJKI52001	70	75	75	90
A1-001	PJKI53002	75	80	90	85
A1-002	PJKI52001	80	65	80	80
A1-002	PJKI52002	75	70	75	90
A1-003	PJKI53003	90	80	70	85
A1-004	PJKI53003	100	75	90	90

Tabel Matakuliah

Kd_MK	Nm_MK
PJKI52001	Konsep Database
PJKI53002	Sistem Operasi
PJKI53003	Sistem Distribusi

# Kebebasan Data (**Data Independence**) \*

- Program-program aplikasi “*diisolasi*” terhadap perubahan-perubahan yang dilakukan terhadap bagaimana data disusun dan disimpan.
- Logical data independence : Proteksi *external schema* terhadap perubahan-perubahan dalam struktur *logical* dari data.
- Physical data independence : Proteksi *conceptual schema* terhadap perubahan-perubahan dalam struktur *physical* dari data.

\* Salah satu manfaat penting dari penggunaan DBMS

# Queries dalam DBMS

- Query : adalah pertanyaan yang melibatkan data yang disimpan dalam suatu RDBMS.  
Contoh queries utk university database :

*Siapa nama mhs dengan nim=A1-1003 ? Select nama from mAHASISWA where NIM= A1-100323456*

*Berapa jumlah mhs yang mengambil kuliah dengan kode mata kuliah kuliah course PJKI52001*

- Query Language : adalah bahasa khusus yang disediakan oleh RDBMS untuk mendefiniskan query seperti : Aljabar Relasi & Kalkulus Relasi.

- Relational Algebra : adalah formal query language yang didasarkan pada sekumpulan "operators" utk memanipulasi data
  - Relational Calculus : formal query language yang didasarkan pada logika matematika

- Data Manipulation Language (DML) adalah bahasa khusus dalam DBMS yang memungkinkan penggunanya utk membuat data, memodifikasi data, dan melakukan query terhadap data.

- Query language adalah subset dari DML
  - DML dan DDL secara kolektif dapat dianggap sebagai data sublanguage bilamana dijadikan satu (*embedded*) dalam suatu "host language" (seperti, C, Java, dll.)

# Concurrency Control

- Eksekusi “user programs” secara serentak (*concurrent*) merupakan aspek yang essensial utk memperoleh kinerja DBMS yang baik.  
Karena akses disk yang begitu sering dan relatif lambat, maka penting utk membuat CPU sibuk dengan mempekerjakannya pada beberapa user programs secara konkuren.
- Tindakan silih-ganti (*interleaving*) pada beberapa user programs yang berbeda dapat menimbulkan terjadinya keadaan yang tidak konsisten  
Seorang user diperbolehkan membaca nilai saldo sebelum seorang user lain selesai melakukan perubahan nilai saldo tersebut.
- DBMS menjamin persoalan inkonsistensi TIDAK akan terjadi akibat eksekusi konkuren, dengan cara membuat sistem seolah-olah para pengguna DBMS sedang menggunakan sebuah “single-user system”

# Transaction : An Execution of a DB Program

- Konsep kunci : transaksi (transaction) adalah suatu urutan tindakan-tindakan basis data (reads/writes) yang bersifat "*atomic*". Setiap transaksi yang dieksekusi secara lengkap harus meninggalkan DB dalam keadaan konsisten jika DB berada dalam keadaan konsisten pada saat transaksi dimulai.
  - Pengguna dpt menentukan beberapa integrity constraints sederhana pada data, dan DBMS akan memaksa constraints tersebut
  - Di luar itu, DBMS tidak akan "mengerti" semantik yang terkandung pada data. Contoh, DBMS tdk mengerti bagaimana bunga bank pada suatu rekening dihitung.
  - Dengan demikian, pada akhirnya jaminan bahwa sebuah transaksi (yang berjalan sendiri) dapat mempertahankan konsistensinya merupakan tanggung jawab dari **pengguna** itu sendiri !

## DATABASE INTEGRITY

### Integrity Constraints

*Constraint* : aturan yang diberikan ke tabel agar data yang dimasukkan valid

Integrity Constraints akan melindungi database dari kerusakan dengan memastikan bahwa perubahan yang dilakukan tidak menyebabkan inkonsistensi data

#### Jenis Integrity Constraint :

1. Inter-relational Constraint (constraint pada satu tabel)  
Meliputi : **Entity Integrity Constraint** dan **Domain Constraint**.
2. Intra-relasional Constraint (constraint melibatkan > 1 tabel)  
Meliputi : **Referential integrity**.
3. Enterprise Integrity (User Defined Integration)

# Scheduling Concurrent Transactions

- DBMS menjamin bhw eksekusi  $\{T_1, \dots, T_n\}$  adalah ekivalen dg beberapa eksekusi serial  $T_1' \dots T_n'$ .
  - Sebelum *reading/writing* sebuah objek, transaksi melakukan permintaan “lock” pada objek, dan menunggu hingga DBMS memberikan lock tsb. Semua locks dilepas pada bagian akhir dari transaksi. (Strict 2PL locking protocol.)
  - Idea dasar: Jika sebuah tindakan dari  $T_i$  (misal, writing X) mempengaruhi  $T_j$  (mungkin reads X), salah satu di antaranya, misal  $T_i$ , akan memperoleh lock pada X lebih dulu dan  $T_j$  dipaksa utk menunggu hingga  $T_i$  selesai. Hal ini pada dasarnya merupakan proses pengurutan dari transaksi
  - Persoalan: Apa yang akan terjadi jika  $T_j$  telah memiliki lock pada Y dan  $T_i$  kemudian melakukan permintaan lock pada Y ? (Deadlock !), untuk ini  $T_i$  atau  $T_j$  harus digugurkan (aborted) dan transaksi diulang kembali (*restarted*) !

**Eksekusi Berurutan dari 2 Transaksi**

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	$PROD\_QOH = 35 + 100$	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	$PROD\_QOH = 135 - 30$	
6	T2	Write PROD_QOH	105

**Masalah Lost Updates**

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T2	Read PROD_QOH	35
3	T1	$PROD\_QOH = 35 + 100$	
4	T2	$PROD\_QOH = 35 - 30$	
5	T1	Write PROD_QOH (Lost update)	135
6	T2	Write PROD_QOH	5

# Jaminan Atomicity

- DBMS menjamin *atomicity (all-or-nothing property)*, termasuk jika sistem mengalami “crash” di tengah-tengah eksekusi suatu transaksi
- **Idea dasar:** Memelihara *log (history)* dari semua tindakan dilakukan oleh DBMS ketika sedang menjalankan sekumpulan transaksi:
  - Sebelum perubahan dilakukan pada suatu basis data, “log entry” yang bersesuaian dipaksa disimpan ke suatu lokasi yang aman. (Write-Ahead Log / WAL protocol)
  - Setelah “*crash*” terjadi, pengaruh-pengaruh dari sejumlah transaksi yang telah dijalankan secara parsial dikembalikan seperti semula (*undone*) menggunakan “log entry” yang telah disimpan secara aman sebelumnya. Namun, jika “*log entry*” tidak disimpan sebelum terjadi crash, perubahan-perubahan yang terjadi tidak diaplikasikan ke basis data.

# The Log

- Tindakan-tindakan yang direkam dalam “Log”:
  - *Ti writes an object:* nilai lama dan nilai baru.
    - “Log record” hrs disimpan ke disk sebelum perubahan dilakukan!
  - *Ti commits/aborts:* “log record” mengindikasikan tindakan ini
- “Log records” disimpan secara berantai berdasarkan “ID” transaksi, sehingga memudahkan untuk melakukan “undo” suatu transaksi tertentu (misalnya, untuk mengatasi “*deadlock*”)
- “Log” biasanya *diduplikasi* dan *diarsipkan* pada media penyimpan yang “stabil”
- Semua aktifitas yang terkait dengan “log” (termasuk semua aktifitas yang terkait dengan *concurrency control* seperti *lock/unlock*, penanganan *deadlocks*, dll.) ditangani secara jelas (*transparent*) oleh DBMS.

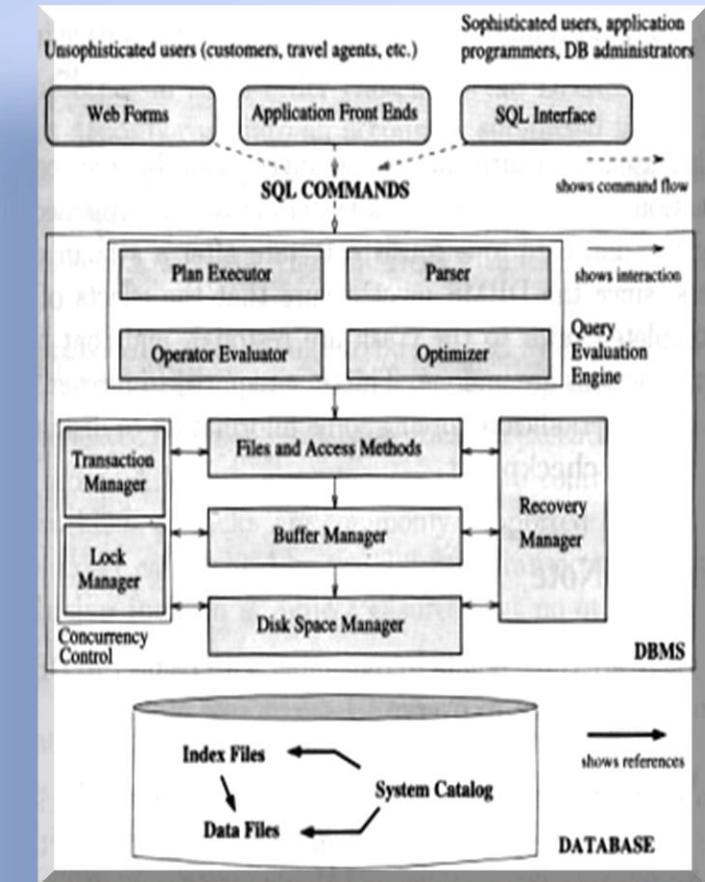
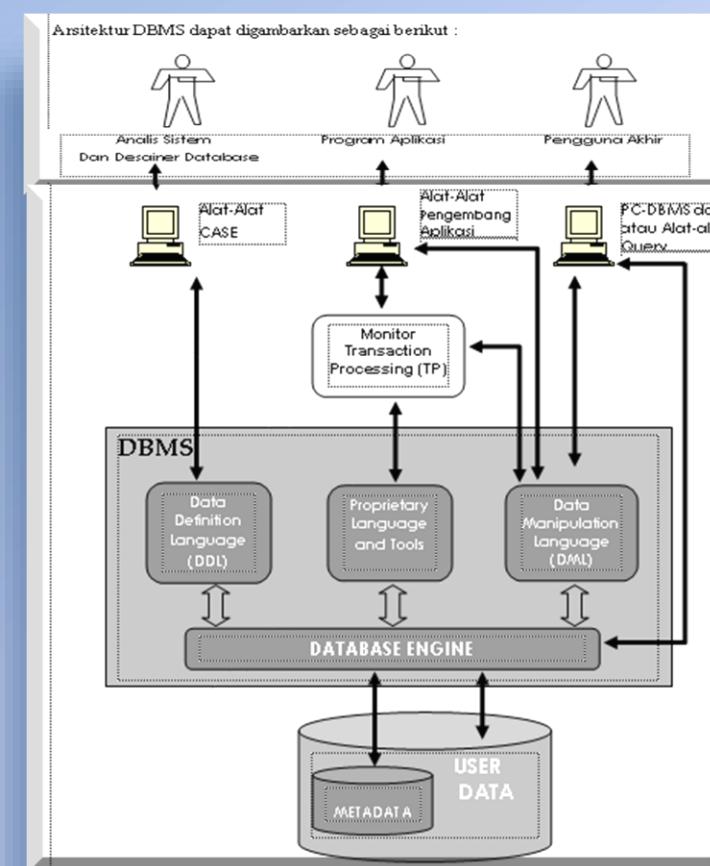
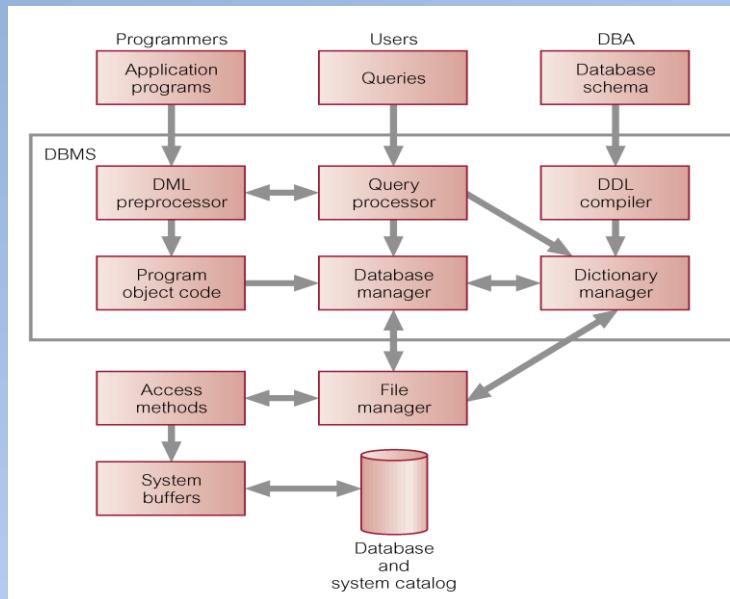
Log Transaksi										
TRL_ID	TRX_NUM	PREV_PTR	NEXT_PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE	
341	101	Null	352	START	****Start Transaction					
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23	
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73	
365	101	363	Null	COMMIT	**** End of Transaction					

↑

TRL\_ID = Transaction log record ID  
TRX\_NUM = Transaction number  
PTR = Pointer to a transaction log record ID  
(Note: The transaction number is automatically assigned by the DBMS.)

# Arsitektur Database : Struktur DBMS

- DBMS memiliki arsitektur berlapis (*layered architecture*) Gambar dibawah ini mengilustrasikan beberapa arsitektur. Masing-masing sistem memiliki variasi sendiri-sendiri.



# Orang-orang yang bekerja dengan Basis Data **(Databases make these folks happy ...)**

- End users & DBMS vendors
- Database Application programmers
- Database Administrator (DBA):
  - Mendesain logical & physical schemas
  - Menangani sekuritas dan otorisasi
  - Bertanggung jawab terhadap ketersediaan data dan pemulihan kembali dari terjadinya kegagalan sistem (*crash recovery*)
  - Melakukan “tuning” basis data sesuai dengan perkembangan yang terjadi
    - tuning basis data adalah peningkatan kinerja pada desain basis data secara fisikal yang mencakup relasi dan view sesuai dengan kebutuhan pengguna

# Rangkuman

- DBMS digunakan untuk memelihara dan melakukan “query” terhadap dataset yang besar
- Manfaat DBMS meliputi pemulihan kembali dari kegagalan sistem, akses serentak, pengembangan aplikasi secara cepat, integritas dan sekuritas data
- Tingkatan abstraksi memberikan kebebasan data
- Suatu DBMS yang khas (typical) memiliki arsitektur berlapis
- DBA mempunyai tanggung jawab yang besar dan *well-paid* !
- DBMS R&D merupakan salah satu area yang luas dan menarik dalam bidang ilmu komputer / informatika

# Tugas : Assignment 1

1. Buat Conceptual Schema beberapa Obyek Tertentu
2. Buat Relationship dari konseptual skema Obyek Tersebut.
3. Buat Database Relational berdasarkan relationship model tersebut, Minimal Menghasilkan 5 tabel yg berelasi.