

PERTEMUAN 1

PENGENALAN DBMS



SEJARAH DBMS

Sejak awal adanya komputer, menyimpan dan memanipulasi data telah menjadi fokus utama sebuah aplikasi. Awalnya DBMS yang didesain oleh Charles Bachman di General Electric pada awal tahun 1960 disebut *Integrated Data Store*. Ini membentuk dasar untuk model data jaringan, yang distandarisasi oleh *Conference On Data System Languages* (CODASYL) dan sangat berarti bagi sistem database pada tahun 1960-an. Bachman adalah penerima penghargaan pertama dari *ACM Turing Award* (penghargaan ilmu komputer setara hadiah Nobel) yang bekerja di bidang database pada tahun 1973.

Definisi

- **Data**

Fakta, teks, hasil pengukuran, gambar, suara, dan video yang bernilai informasi.

- **Informasi**

Data yang telah diproses sebagai bahan dalam proses pengambilan keputusan.

- **Database**

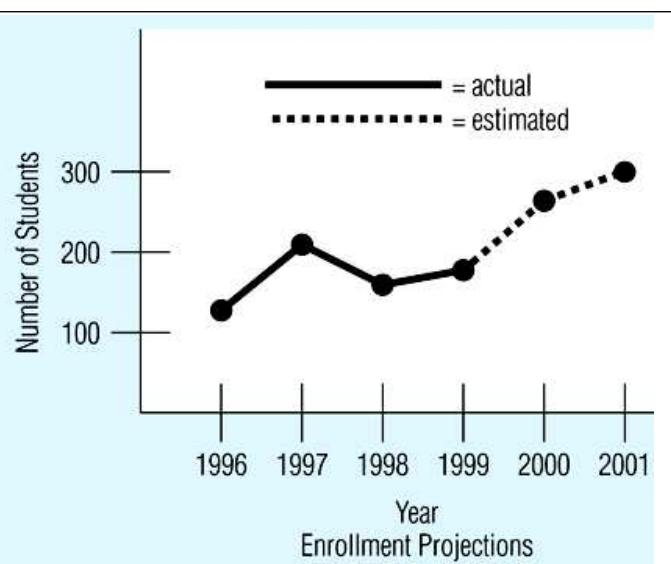
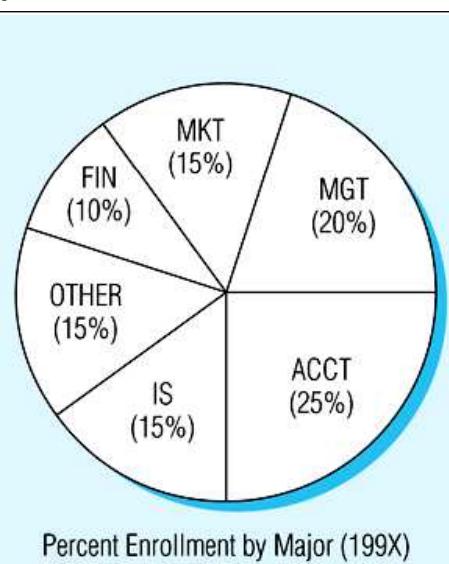
Kumpulan data yang terorganisir berdasarkan suatu struktur hubungan.

- **Metadata**

Data yang mendeskripsikan data lain.

Informasi

Informasi - dapat dimanfaatkan sebagai dasar untuk pengambilan keputusan dan memahami permasalahan/situasi



Metadata

Deskripsi tentang format dan karakteristik data, termasuk tipenya, ukurannya, nilai-nilai yang absah, dan dokumentasi lainnya.

Data Item		Value			
Name	Type	Length	Min	Max	Description
Course	Alphanumeric	30			Course ID and name
Section	Integer	1	1	9	Section number
Semester	Alphanumeric	10			Semester and year
Name	Alphanumeric	30			Student name
ID	Integer	9			Student ID (SSN)
Major	Alphanumeric	4			Student major
GPA	Decimal	3	0.0	4.0	Student grade point average

Evolusi Teknologi Database

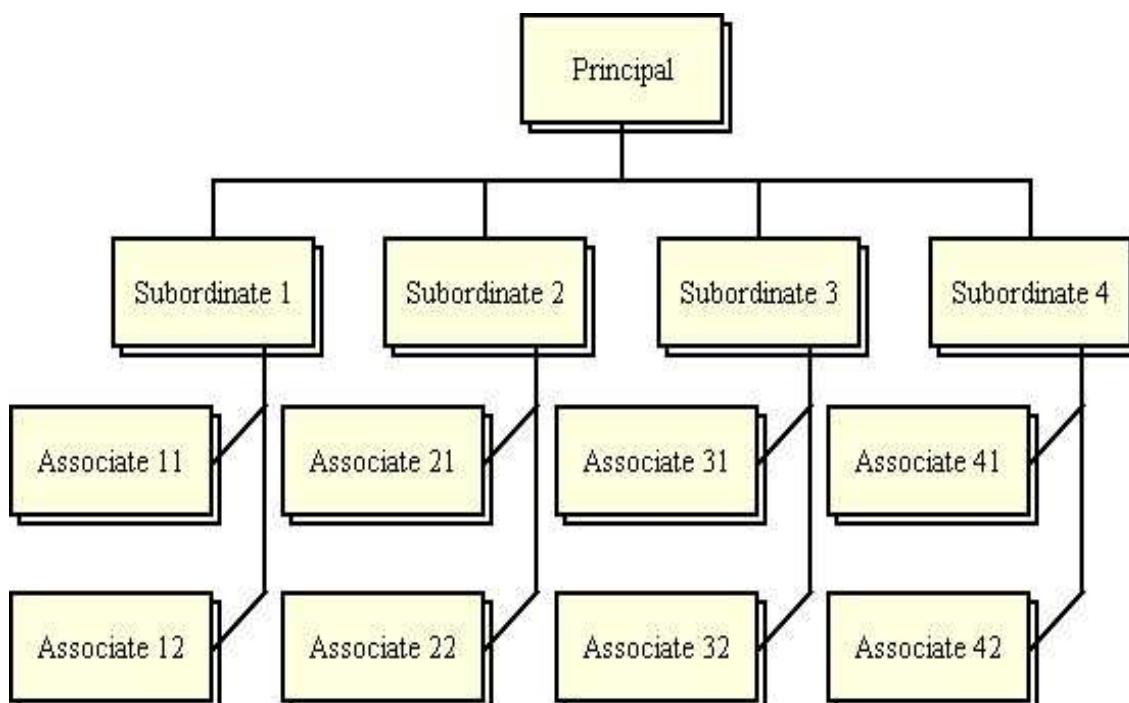
- **Flat File** → 1960an - 1980an
- **Hierarchical** → 1970an - 1990an
- **Network** → 1970an - 1990an
- **Relational** → 1980an - sekarang
- **Object-oriented** → 1990an - sekarang
- **Object-relational** → 1990an - sekarang

Database Flat File

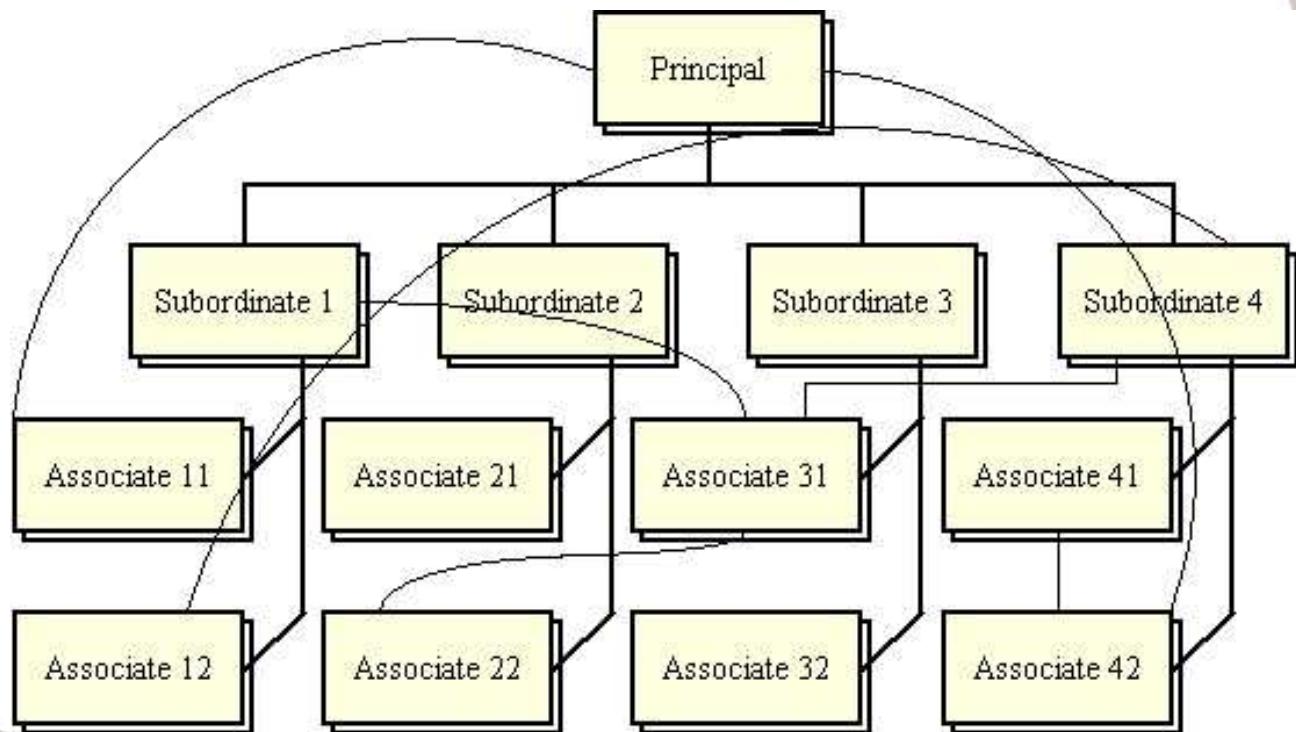
Database *flat file* sama dengan file data pada *spreadsheet* (misal MS Excel™), berupa satu file berisi baris-baris dengan jumlah kolom tetap yang disimpan berurutan dalam file.

NIP	Nama	Nama Depan	Telp
123-45-6789	Malik	Abdul	021-555-1234
987-65-4321	Silalahi	Roy	022-543-9876
987-65-4321	Mariano	Arie	021-234-5678
567-89-0123	Iskandar	Dony	021-987-6431

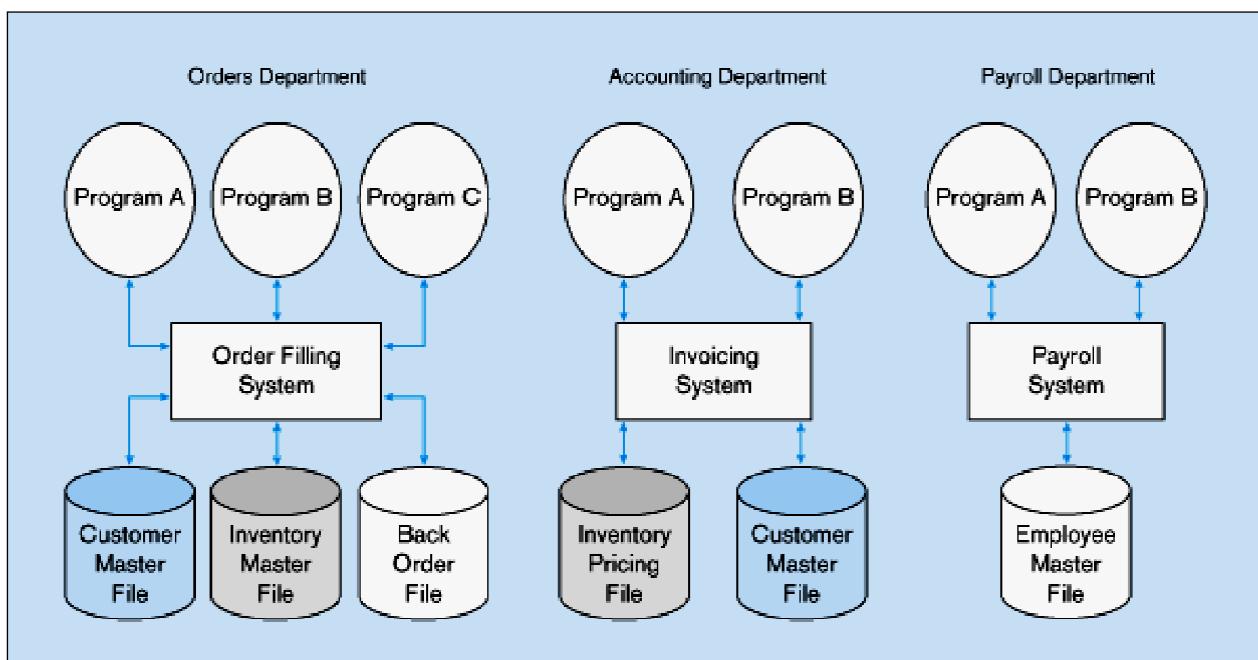
Database Hierarchical



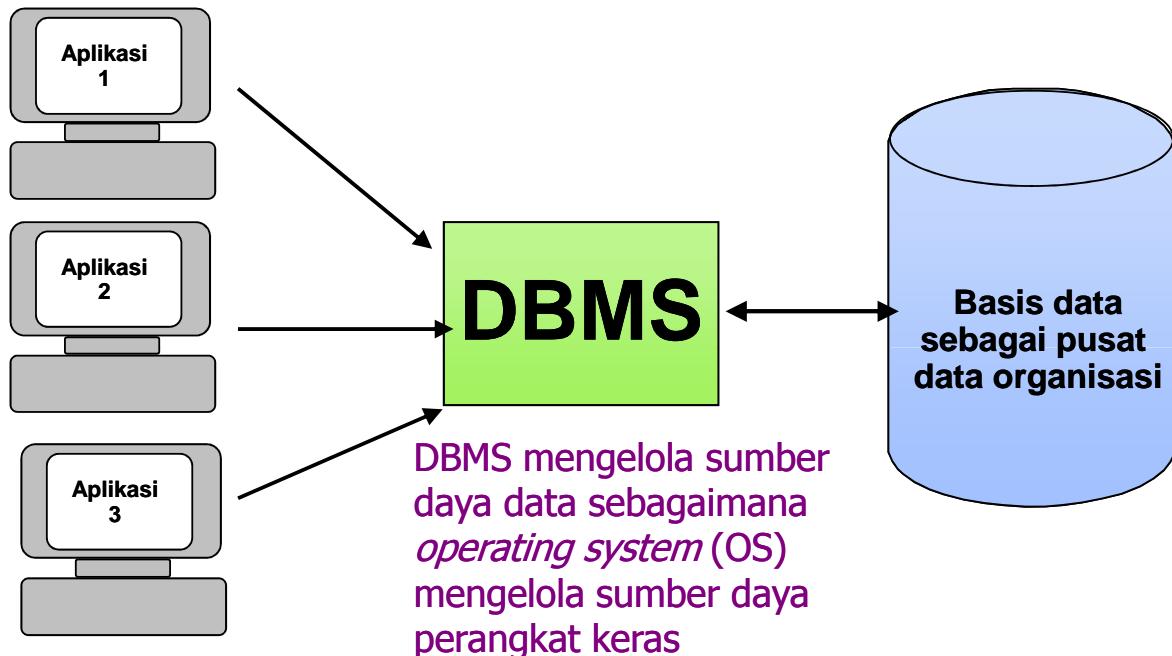
Database Network



Sistem Berbasis File



DBMS



Sistem File Versus DBMS

- Kapasitas penyimpanan data
- Kemampuan mengakses data
- Konkurensi data
- Perlindungan data dari kegagalan
- Keamanan data

Manfaat DBMS

- Independensi data
- Akses data efisien
- Integritas dan keamanan data
- Administrasi data
- Akses konkuren dan Crash Recovery
- Waktu pengembangan aplikasi terkurangi



TINGKAT ABTRAKSI DALAM DBMS

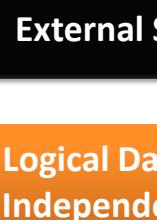
1. Model Relasional
2. Tingkat Abstraksi dalam DBMS
3. Independensi Data

- Dalam membangun deskripsi tentang pusat data dalam model adalah suatu hubungan, yang dapat dianggap sebagai kumpulan record.
- Deskripsi data dalam istilah model data disebut skema. Dalam model relasional, skema digunakan untuk relasi menentukan nama, nama setiap field (atau atribut atau kolom), dan tipe setiap field.

- Independensi data adalah program aplikasi yang telah terisolasi dari perubahan dalam struktur data yang disimpan.
- Independensi data dicapai melalui penggunaan dari tiga tingkat data abstraksi ; khususnya, skema konseptual dan skema eksternal yang mempunyai manfaat yang berbeda pada bidang ini.

Data dalam sebuah DBMS dibagi menjadi tiga tingkatan yaitu:

- Skema Konseptual
- Skema Fisik
- Skema Eksternal



Logical Data Independen

External Schema 1

External Schema 2

External Schema 3

Conceptual Schema

Physical Data Independen

Physical Schema

DISK

QUERY DALAM DBMS

- Query berfungsi untuk memudahkan mendapatkan informasi dari database yang nantinya akan digunakan untuk menentukan nilai informasi bagi seorang pengguna

- **Queries:** pertanyaan yang melibatkan data di dalam DBMS.
- **Bahsa Query:** bahasa khusus yang dapat digunakan untuk menampilkan query yang diinginkan.
- **Kalkulus relasional:** adalah bahasa query formal berdasarkan logika matematika, dan query yang ditampilkan dalam bahasa ini memiliki arti dan maksud yang tepat.
- **Aljabar relasional:** merupakan bahasa query formal berdasarkan kumpulan operator untuk memanipulasi relasi, yang setara dengan kalkulus.

Independensi Data

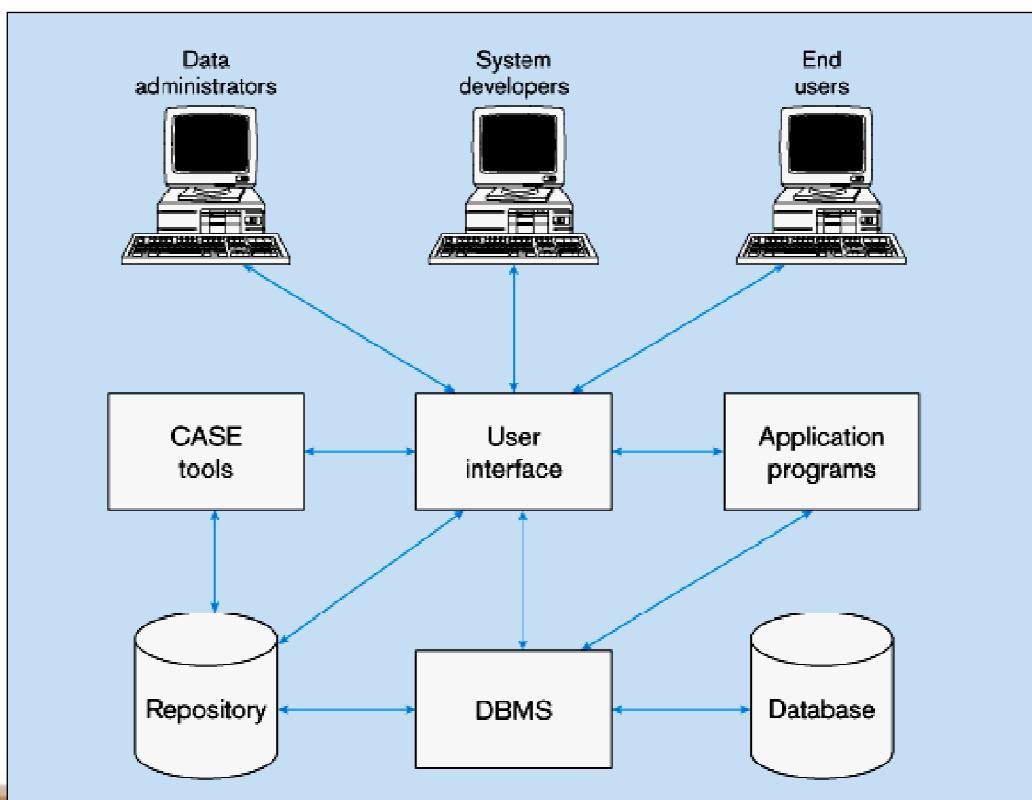
- Independensi data adalah program aplikasi yang telah terisolasi dari perubahan dalam struktur data yang disimpan.
- Independensi data dicapai melalui penggunaan dari tiga tingkat data abstraksi ; khususnya, skema konseptual dan skema eksternal yang mempunyai manfaat yang berbeda pada bidang ini.

Komponen Sistem Database

- **Repositori** → Pusat penyimpanan metadata.
- **Database Management System (DBMS)** → Perangkat lunak untuk mengelola database.
- **Database** → Pusat penyimpanan data.
- **Program Aplikasi** → Perangkat lunak pengguna data.
- **User Interface** → Fasilitas interaksi antara pengguna dan data secara tekstual atau grafis.
- **CASE Tools** → Computer - Aided Software Engineering.

- **Administrator Data** → Personil yang bertanggung-jawab memelihara database.
- **Developer Sistem** → Personil yang bertanggung-jawab merancang program aplikasi beserta struktur datanya dalam database.
- **End User** → Orang yang menggunakan aplikasi dan database.

Komponen Sistem Database



Model Data

- **Model Data**

Kumpulan konstruksi deskripsi data level tinggi yang menyembunyikan banyak detail penyimpanan level rendah. DBMS memungkinkan pengguna untuk menentukan data yang disimpan dalam model data.

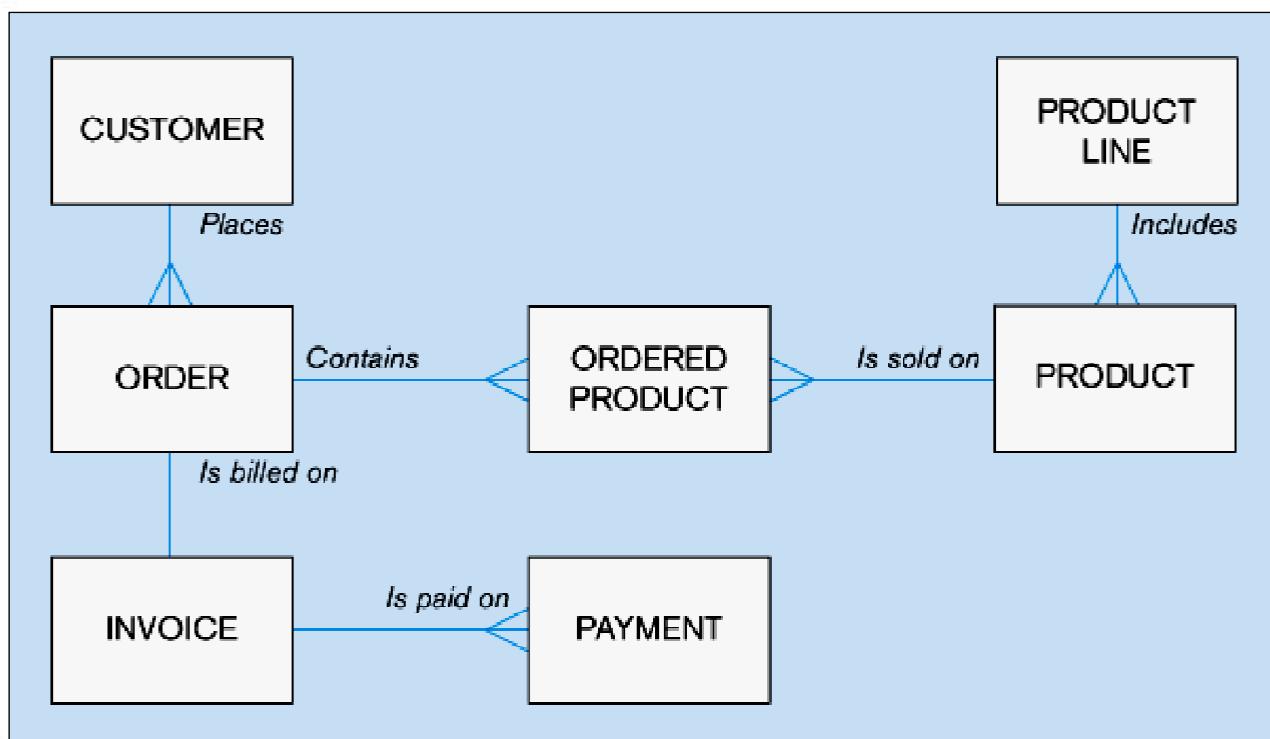
- **Skema Logis**

Mendeskripsikan data yang disimpan dalam model data DBMS berupa semua relasi yang disimpan dalam database.

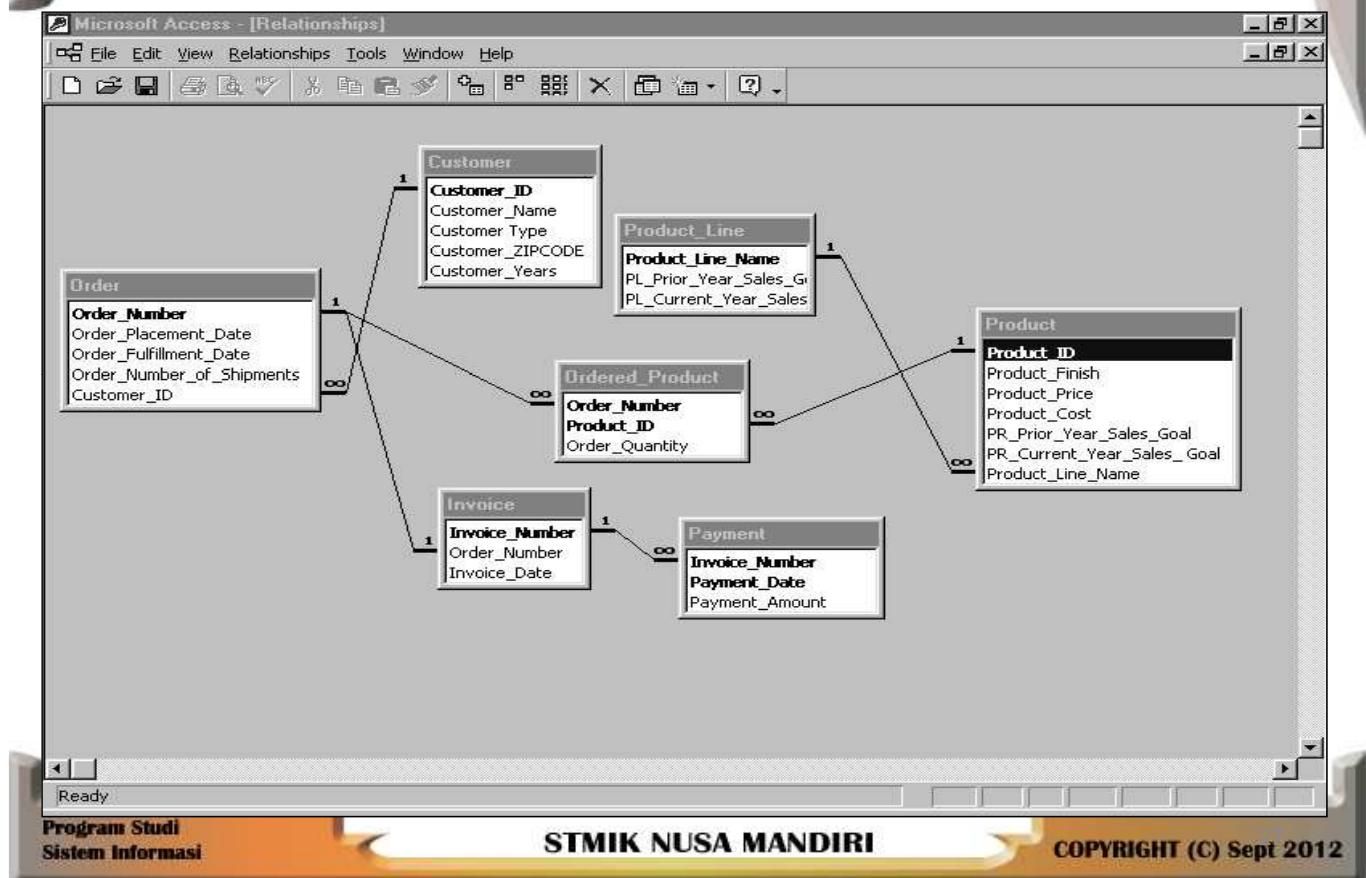
- **Skema Fisik**

Menentukan detail penyimpanan data berupa mendeskripsikan bagaimana relasi dalam skema Logis disimpan pada alat penyimpanan sekunder seperti disk atau tape.

Contoh Skema Logis



Contoh Skema Fisik



PERTEMUAN 2

MODEL DATA

MODEL ENTITY RELATIONSHIP (MODEL E-R)

(Chap. 2 – Ramakrishnan)

Tujuan

- Menjelaskan konsep model data Entity-Relationship (ER)
- Menjelaskan peran model data ER sebagai tahap awal proses perancangan basisdata

Topik

- Model Data
- Model Entity-Relationship
 - Apa model ER
 - Mengapa model ER
 - Konsep model ER
 - Skema dan Instance
 - Skema model ER
 - Diagram ER
 - Notasi Diagram ER
 - Contoh Diagram ER
 - Komentar mengenai model ER

DESAIN DATABASE

Proses desain database dapat dibagi menjadi enam langkah

1. Analisis Persyaratan : Langkah yang sangat penting dalam merancang aplikasi database adalah untuk memahami data apa yang akan disimpan dalam database
2. Database Desain Konseptual: Informasi yang dikumpulkan secara konseptual.
3. Desain Database Logical: memilih sebuah DBMS untuk desain, dan mengubah desain dari bentuk konseptual ke skema database dalam data model DBMS

4. Skema perbaikan : menganalisis koleksi hubungan dalam skema relasional database
5. Database Desain Fisik: beban kerja database yang melibatkan indeks pada beberapa tabel dan clustering
6. Keamanan Desain: mengidentifikasi bagian-bagian dari database yang dapat mengakses sebuah DBMS

Model Data

- Model data adalah kumpulan konsep yang digunakan untuk menjelaskan struktur basis data, yang meliputi:
 - hubungan (*relationship*) antar data
 - arti (*semantic*) data
 - batasan (*consistency constraint*) data
 - representasi data

MODEL DATA

Model data adalah kumpulan perangkat konseptual data tingkat tinggi yang menyembunyikan detail tentang bagaimana data disimpan.

Model data semantik adalah model data abstrak yang membuatnya lebih mudah bagi pengguna untuk memulai dengan deskripsi awal yang baik dari data dalam suatu organisasi. Model ini mengandung berbagai macam konsep yang menggambarkan susunan aplikasi nyata.

Model Relasional adalah model data yang menggambarkan data dengan relasinya.

SKEMA

- Deskripsi data dalam istilah model data disebut skema
- Dalam model relasional, skema untuk suatu relasi menentukan nama dari setiap field (atribut atau kolom), dan jenis dari masing-masing field.
- contoh, informasi mahasiswa dalam sebuah database universitas dapat disimpan dalam suatu relasi dengan skema sebagai berikut:

Mahasiswa(sid: **string**, nama: **string**, login: **string**, umur: **integer**, IPK: **real**)

Model Data

- Macam-macam Model Data
 - *Object-based Logical Model*
 - Misalnya: *Entity-Relationship (ER)*, *Object-oriented*, *Semantic*, dan *Functional Data Model*
 - *Record-based Logical Model*
 - Misalnya: *Relational*, *Hierarchy*, dan *Network Data Model*

Model ER

- Apakah Model ER?
 - Model data konseptual
 - Tidak mendeskripsikan cara data disimpan didalam komputer
 - Mendekati pengamatan pemakai terhadap data riil
 - Digunakan pada tahap awal perancangan basisdata

MODEL E-R (Entity Relationship)

Model E-R menggambarkan data yang terlibat dalam organisasi, hubungan objek serta dapat digunakan untuk mengembangkan desain awal database

Model ER sangat penting terutama perannya dalam desain database. Model ER menyediakan konsep yang memungkinkan untuk berpindah dari deskripsi apa yang pengguna inginkan pada database , untuk menjelaskan lebih rinci dan dapat diimplementasikan dalam DBMS.

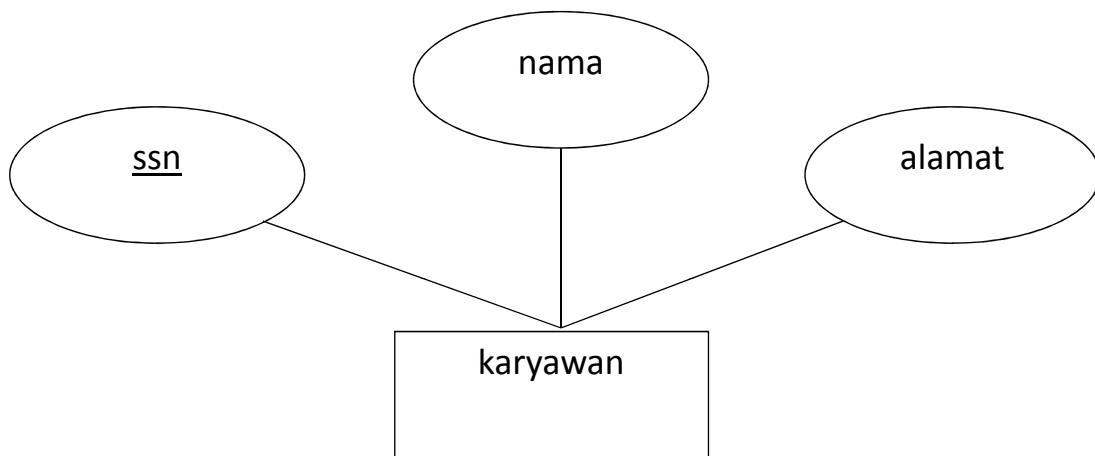
Model ER

- ◆ Mengapa Model ER?
 - ◆ *High level*
 - ◆ Mudah dimengerti karena tidak termasuk perincian implementasi
 - ◆ Sebagai alat komunikasi dengan pemakai nonteknis lain
 - ◆ Untuk memastikan apakah semua kebutuhan data pemakai sudah tercakup dan tidak ada konflik
 - ◆ Bagaimana data disimpan bisa diabaikan untuk sementara

ENTITAS, ATRIBUT, DAN ENTITY SET

- Model ER didasarkan atas *entity* dan hubungan antar *entity*, dimana
 - *entity*:
 - Objek riil yang dapat dibedakan satu sama lain
 - Dapat berupa objek yang ada secara fisik (orang, mobil, ...)
 - Dapat berupa objek yang ada secara konsep (organisasi, pekerjaan, mata kuliah, ...)

HIMPUNAN ENTITAS KARYAWAN



- Himpunan entitas diwakili oleh persegi panjang.
- Atribut diwakili oleh oval.
- Atribut kunci primer digarisbawahi.

– *attribute*:

- Sifat-sifat yang dimiliki oleh entity
- Contoh: entity Karyawan, attribute: nama, alamat, umur, ...

– *relationship*:

- Hubungan antar entitas melalui atribut pada suatu entitas yang merujuk ke entitas lainnya
- Hubungan dapat dinyatakan secara implisit, atau eksplisit
- Hubungan memiliki batasan-batasan, yaitu: rasio hubungan, dan partisipasi hubungan

Model ER

- *Schema, Instance*
 - Model data memisahkan deksripsi data dari datanya sendiri
 - Deskripsi data disebut skema (*schema, intension*)
 - Data didalam basisdata pada suatu saat disebut instance (*extension*)
 - Data didalam basisdata dapat berubah setiap saat, akan tetapi skema lebih bersifat statis

Model ER

- Contoh *Schema, dan Instance*

Schema: EMPLOYEE(Name, Age, Salary)

COMPANY(Name, Headquarter, President)

Instance: EMPLOYEE

Budi	55	80k
Lisa	40	30k
Martin	25	20k

COMPANY

Bimoli	Depok	Budi
IDM	Bogor	Bob

Model ER

- Schema model ER terdiri atas
 - Entitas (*entity*)
 - Attribute & Nilainya (*attribute value sets*)
 - Attribute kunci (*key attribute*)
 - Hubungan (*relationship*)

Model ER

- Entitas (*entity*)
 - Tipe entitas (*entity type, entity set*)
 - Adalah kumpulan entitas yang memiliki atribut-
atribut yang sama
 - Entitas lemah (*weak entity*)
 - Adalah entitas yang tidak mempunyai cukup
atribut untuk membentuk suatu *key* yang unik

Model ER

- Macam-macam atribut:
 - *Simple/Atomic*: atribut yang tidak dapat dibagi, eg. Gaji
 - *Composite*:
 - Atribut yang dapat dibagi menjadi beberapa atribut dasar
 - eg. Nama (First_Name, Last_Name), Alamat (Street, Number, City)
 - *Single-valued*: hanya memiliki satu harga, eg. Gaji
 - *Multi-valued*: memiliki lebih dari satu harga, biasanya punya batas bawah dan batas atas, eg. Gelar

Model ER

- *Derived*:
 - nilainya diturunkan dari atribut lain yang disimpan (*stored attribute*), eg. Umur diturunkan dari atribut: Tgl_lahir
- *Null-valued*
 - Suatu nilai khusus bagi suatu atribut
 - Digunakan apabila nilai suatu atribut dari suatu entitas
 - Tidak diketahui apakah nilai dari atribut tsb ada atau tidak
 - Memiliki nilai, hanya tidak diketahui nilainya (*missing*)
- *Value set (domain)* dari atribut:
 - Adalah kumpulan nilai/harga yang dapat dimiliki oleh atribut suatu entitas

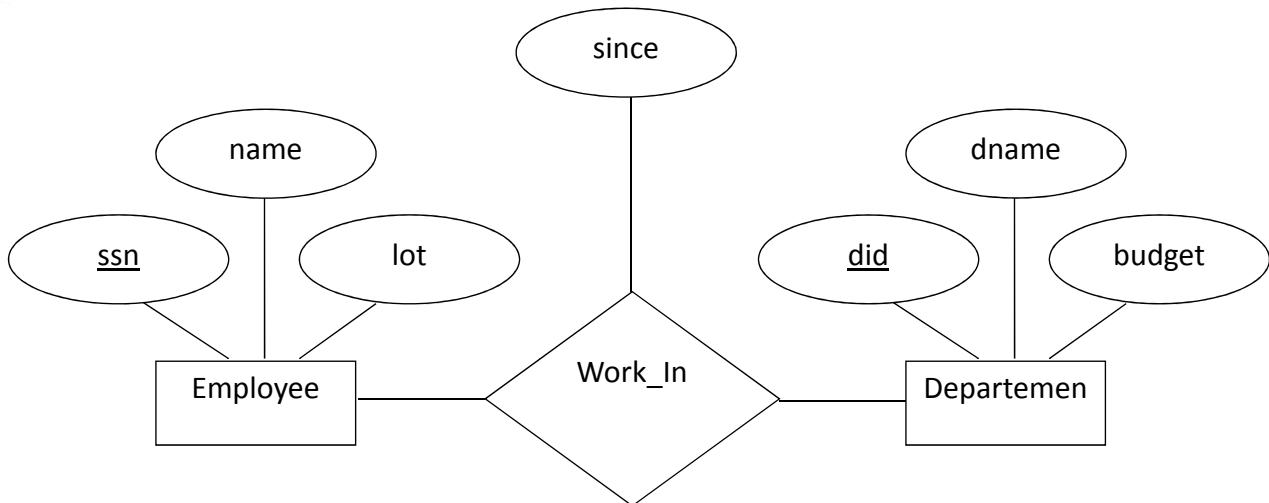
Model ER

- Atribut kunci
 - Atribut yang unik dari suatu entitas
 - Berfungsi sebagai identitas
 - Suatu entitas dapat lebih dari satu, atau berupa composite attribute
- Atribut kunci bagi **weak entity**
 - Hanya bersifat parsial.
 - Atribut kunci yang sesungguhnya merupakan gabungan dari atribut kunci entitas kuat dan atribut kunci parsial dari entitas lemahnya

Model ER

- *Relationship Type*
 - Kumpulan hubungan antar entitas-entitas tertentu
 - Disebut berhubungan apabila ada satu atribut dari suatu entitas merujuk ke atribut entitas lainnya, contoh:
 - Atribut Manager dari entitas DEPARTMENT merujuk ke entitas EMPLOYEE yang memimpin suatu departemen
 - Atribut ControllingDepartment dari entitas PROJECT merujuk ke entitas DEPARTMENT yang menangani suatu proyek
 - Atribut Supervisor dari entitas EMPLOYEE merujuk ke EMPLOYEE yang menjadi pengawas karyawan ybs.

HIMPUNAN RELASI Work_In



Pada gambar diatas membahas tentang relasi yang saling berhubungan dimana setiap relasi meliputi divisi dengan para pekerjanya. Catatan untuk relasi tersebut mungkin termasuk kedalam kumpulan entitas yang sama. Sebagai contoh, kita mungkin bisa mengatur sekumpulan relasi antara para pekerja dengan divisinya

PERTEMUAN 3

Model E-R (Lanjutan)

- *Relationship Constraints*
 - Merupakan kendala yang membatasi kemungkinan kombinasi entitas yang terlibat dalam *relationship instance*
 - Contoh: setiap pegawai hanya bekerja untuk satu departemen
- *Jenis Relationship Constraints*
 - *Cardinality ratio* (kardinalitas)
 - *Participation Constraint*

1 : 1

- *Cardinality Ratio*
 - Menunjukkan banyaknya *relationship instance* dimana entitas dapat berpartisipasi kedalamnya
 - Jenisnya: 1:1, 1:N, dan M:N
 - Contoh: DEPARTMENT: EMPLOYEE adalah 1:N

Model ER

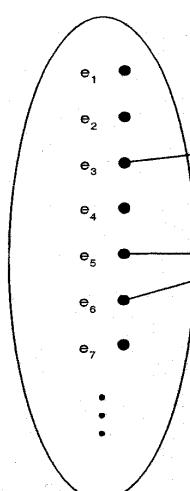
◆ *Participation Constraint*

- ◆ Menunjukkan batasan keterlibatan suatu entitas dalam suatu relationship
- ◆ Jenisnya: *Total*, dan *Partial*
- ◆ *Total*: setiap entitas harus terlibat dalam suatu relationship
- ◆ *Partial*: tidak harus seluruh entitas terlibat dalam suatu relationship

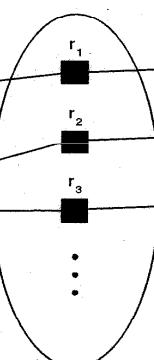
Model ER

1 : 1

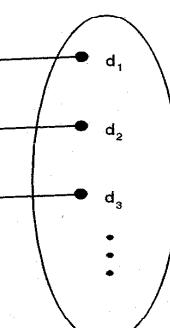
EMPLOYEE



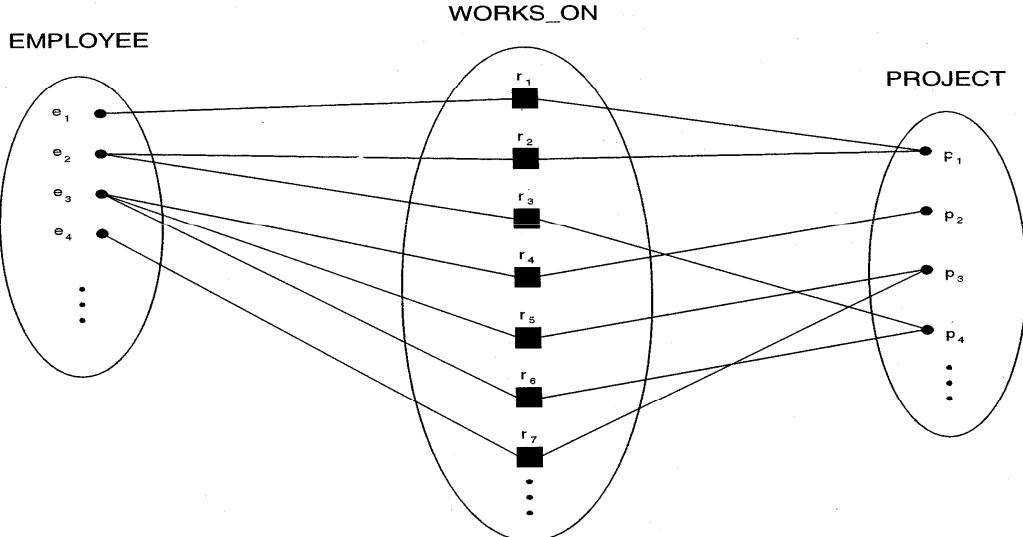
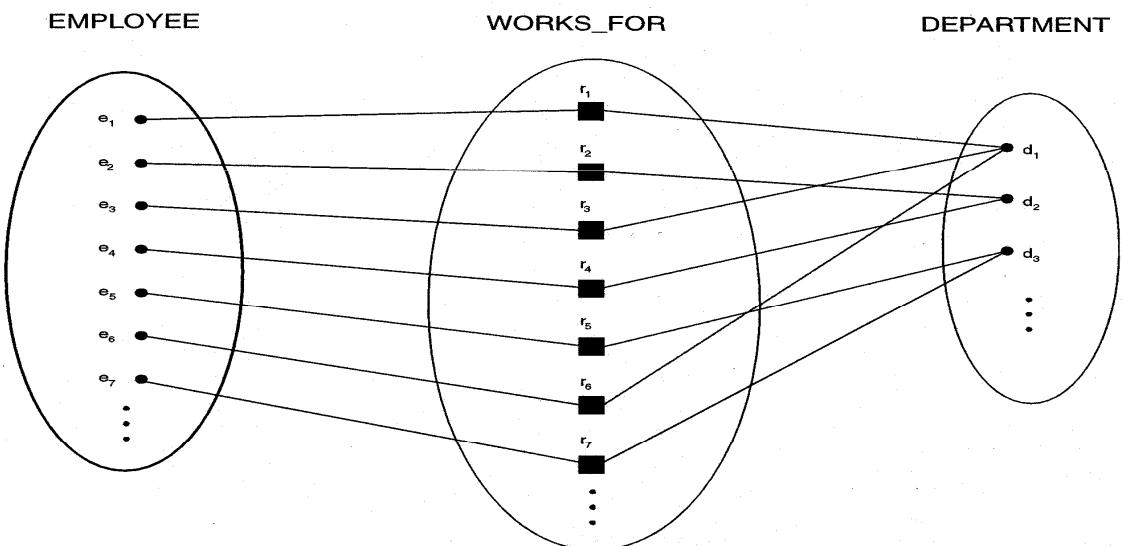
MANAGES



DEPARTMENT



M : 1

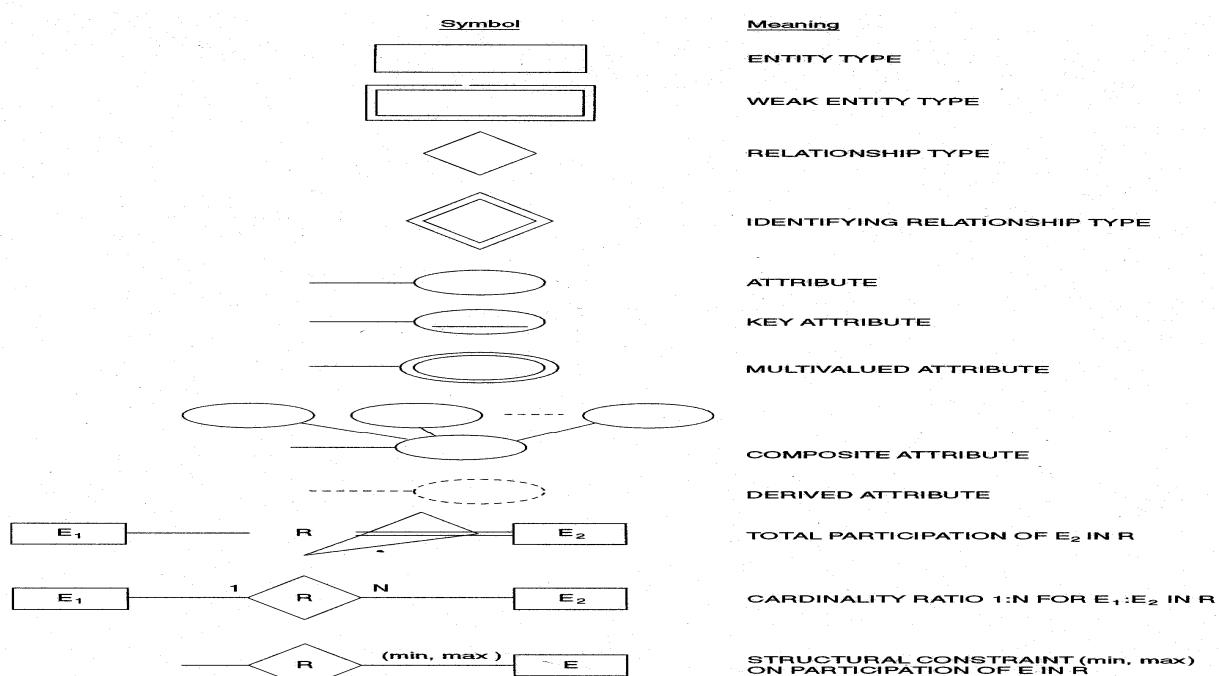


- *Attribute dari Relationship type*
 - Memiliki attribute yang menunjukkan sifat-sifat yang dimiliki oleh relationship type
 - Contoh :
 - WORKS_ON
 - Attribute HOURS untuk menyimpan data lamanya karyawan bekerja untuk suatu proyek permimpin
 - MANAGES
 - Attribute STARTDATE untuk menyimpan data kapan seorang employee menjadi manager

DIAGRAM ER

- DIAGRAM ER
 - Merepresentasikan skema ER secara diagram
 - Ada alat bantu untuk model ER (CASE TOOLS) :
 - Analisa
 - Menggambar
 - Mapping ke data model lain

NOTASI DIAGRAM ER



Case study CASE 1: Perancangan Basisdata COMPANY

- Phase-1: Deskripsi mengenai COMPANY
 - 1 --
 - Perusahaan memiliki beberapa departemen.
 - Setiap departemen memiliki nama dan kode departemen yang *unique*, dan seorang pegawai yang *me-manage* departemen tsb.
 - Database menyimpan tanggal mulai bekerja dari setiap Manager departemen
 - Sebuah departemen dapat memiliki beberapa lokasi

Case study...

-- 2 --

- ◆ Sebuah departemen mengawasi sejumlah proyek.
- ◆ Setiap proyek memiliki nama yang unik, kode yang unik, dan sebuah lokasi

Case study...

-- 3 --

- Bagi setiap pegawai, database menyimpan nama, ssn, alamat, gaji, jenis_kelamin, dan tanggal_lahir.
- Seorang pegawai ditugaskan pada sebuah departemen, namun dapat bekerja untuk beberapa proyek yang tidak selalu berada dibawah pengawasan departemen yg sama.
- Database menyimpan data jumlah jam kerja per minggu setiap pegawai pada setiap proyek yang dikerjakannya.
- Database menyimpan data atasan langsung setiap pegawai

Case study...

-- 4 --

- ◆ Untuk keperluan asuransi, database juga menyimpan data tanggungan (*dependents*) setiap pegawai
- ◆ Atribut setiap tanggungan: Name, Sex, BirthDate, Relationship

Case study...

- ◆ Phase-2: Rancangan Konseptual Awal

(1). Entity Type: DEPARTMENT

- ◆ Atribut: Name, Number, {Locations}, Manager, ManagerStartDate
- ◆ Atribut Locations: multi-valued atribut

(2). Entity Type: PROJECT

- ◆ Atribut: Name, Number, Location, ControllingDepartment

Case study...

(3). Entity Type: EMPLOYEE

- ◆ Atribut: Name(FName, MInit, LName), SSN, Sex, Address, Salary, BirthDate, Department, Supervisor, {WorksOn(Project, Hours)}
- ◆ Atribute Name, dan Address dapat merupakan Composite attribute (harus dicek kembali dengan kebutuhan pemakai)

(4). Entity Type: DEPENDENT

- ◆ Atribut: Employee, DependentName, Sex, BirthDate, Relationship

Case study...

♦ Implicit relationship:

- ◆ Atribut Manager (DEPARTMENT) menunjuk ke Pegawai yang menjadi Manager suatu departemen
- ◆ Atribut ControllingDepartment (PROJECT) menunjuk ke departemen yang mengawasi proyek tsb
- ◆ Atribut Supervisor(EMPLOYEE) menunjuk ke pegawai yang menjadi pengawas pegawai tsb

Case study...

- ◆ Penghalusan Rancangan Konseptual:
 - ◆ MANAGES:
 - ◆ Relasi antara EMPLOYEE dengan DEPARTMENT
 - ◆ Cardinality Ratio= 1:1
 - ◆ Partisipasi EMPLOYEE: parsial
 - ◆ Partisipasi DEPARTMENT: tidak dinyatakan secara jelas oleh hasil dari Phase-1
 - ◆ Perlu ditanyakan kepada pemakai: adakah yang menyatakan bahwa suatu departemen harus selalu memiliki Manager
 - ◆ Tipe Relasi MANAGES memiliki atribut: StartDate

Case study...

- ◆ WORKS_FOR:
 - ◆ Relasi antara DEPARTMENT dengan EMPLOYEE
 - ◆ Cardinality Ratio= 1:N
 - ◆ Partisipasi EMPLOYEE: total
 - ◆ Partisipasi DEPARTMENT: total

Case study...

◆ CONTROLS:

- ◆ Relasi antara DEPARTMENT dengan PROJECT
- ◆ Cardinality Ratio= 1:N
- ◆ Partisipasi DEPARTMENT: ditetapkan sebagai parsial (sesudah konsultasi dengan para pemakai)
- ◆ Partisipasi PROJECT: total

Case study...

◆ SUPERVISION:

- ◆ Relasi antara EMPLOYEE (peran yang diawasi) dengan EMPLOYEE (peran Supervisor)
- ◆ Cardinality Ratio= 1:N
- ◆ Partisipasi kedua EMPLOYEE: parsial
- ◆ Hasil wawancara: Tidak semua pegawai adalah Supervisor, dan tidak semua pegawai memiliki Supervisor

Case study...

◆ WORKS_ON:

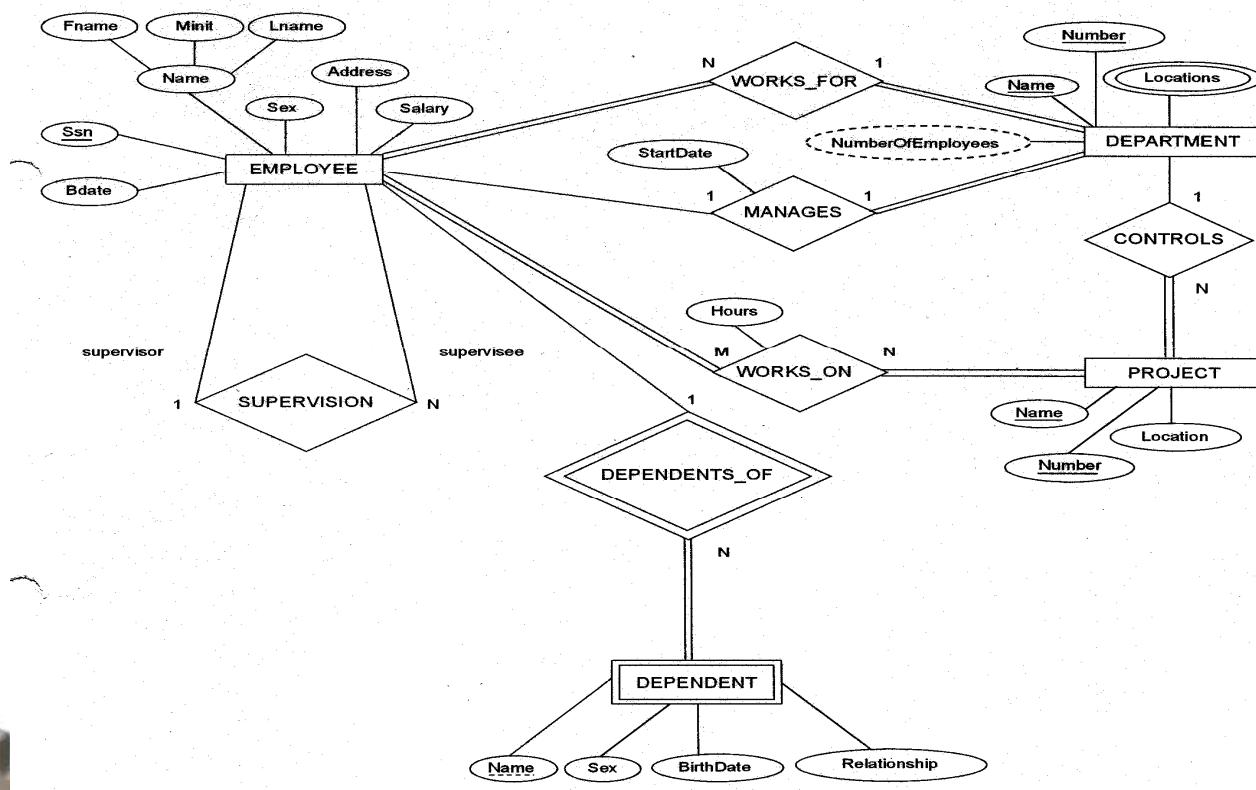
- ◆ Relasi antara EMPLOYEE dengan PROJECT
- ◆ Cardinality Ratio= M:N
- ◆ Partisipasi EMPLOYEE: total
- ◆ Partisipasi PROJECT: total
- ◆ Hasil wawancara: Sebuah proyek dapat memiliki beberapa pegawai yang bekerja didalamnya.

Case study...

◆ DEPENDENTS_OF:

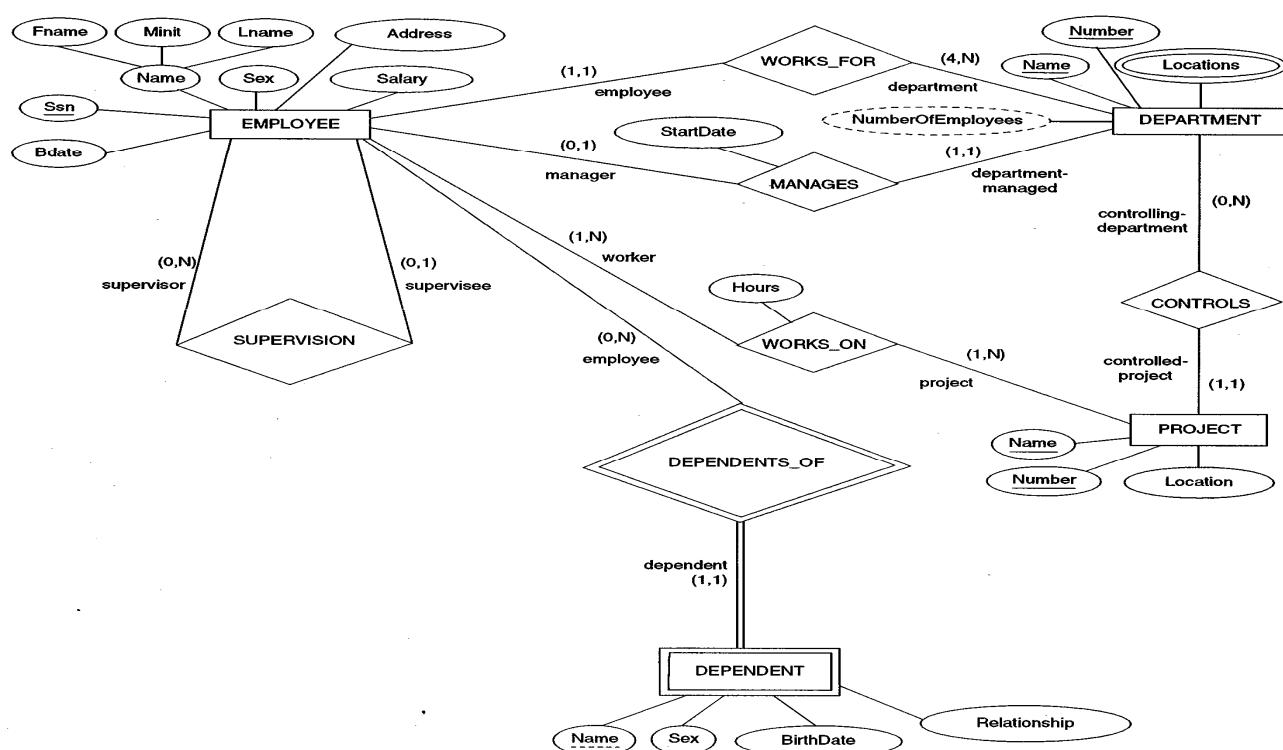
- ◆ Relasi antara EMPLOYEE dengan DEPENDENT
- ◆ Cardinality Ratio= 1:N
- ◆ DEPENDENT adalah sebuah *weak entity*
- ◆ Partisipasi EMPLOYEE: parsial
- ◆ Partisipasi DEPENDENT: total

Case study...



Case study...

- Diagram ER skema COMPANY (Alternatif)



PERTEMUAN 4

Model Data Relational

(Chap. 4 – Conoly)

(Hal 51-83 – Ramakrishnan)

Tujuan

- Menjelaskan model data relasional
- Menjelaskan langkah-langkah pemetaan dari diagram ER ke model data relasional

Topik

- asal-usul dari model relasional.
- terminologi dari model relasional.
- bagaimana tabel digunakan untuk merepresentasikan data
- sifat hubungan database.
- Bagaimana mengidentifikasi *candidate, primary, alternate, and foreign keys*.

Model Data Relasional

- E.J. Codd, 1970
- Merupakan model data logikal yang populer
- Secara intuitif dapat dilihat sebagai data yang disimpan dalam sekumpulan tabel-tabel dua dimensi, dengan sifat khusus
- Konsep tabel ~ konsep “relation” pada matematik
- Tabel terdiri atas sekumpulan kolom lengkap dengan namanya dan sejumlah baris yang tidak bernama

- ◆ Setiap kolom memiliki domain tertentu
- ◆ Hubungan antar tabel dinyatakan secara eksplisit dengan duplikasi kolom dari satu tabel pada tabel lain
- ◆ Setiap relasi harus memiliki sebuah kolom atau gabungan kolom yang memberikan identifikasi unik untuk setiap baris dari relasi. Identitas ini sering disebut *key attribute* (atribut kunci)

- ◆ Tabel/Relasi/Relation:
 - ◆ Sebuah bentuk pernyataan data secara grafis dua dimensi yang terdiri dari sekumpulan kolom bernama dan sejumlah baris
 - ◆ Relasi adalah tabel dengan kolom dan baris.
- ◆ Baris/Tuple:
 - ◆ Baris-baris yang ada didalam sebuah tabel yang menyatakan isi dari tabel tersebut
- ◆ Attribute:
 - ◆ Suatu dikaitkan adalah kolom bernama relasi.

- ◆ Tupel adalah panah relasi.
- ◆ Derajat relasi adalah jumlah atribut yang dikandungnya.
- ◆ Kardinalitas relasi adalah jumlah tuple yang dikandungnya.
- ◆ *Domain*:
 - ◆ Kumpulan nilai-nilai yang berlaku untuk sebuah kolom dari sebuah tabel
 - ◆ Setiap atribut dalam relasi didefinisikan pada **domain**.
 - ◆ Setiap kolom mempunyai *domain* tertentu dan beberapa kolom dapat mempunyai sebuah *domain* yang sama. Contoh: telp_rumah, telp_kantor

- ◆ *Relation instance*:

- ◆ Kumpulan baris-baris dari relasi yang masing-masing terdiri dari nilai-nilai tertentu yang menyatakan nilai dari informasi yang disimpan pada saat tertentu

Attribute

TEMAN	No	Nama	Telp_r	Telp_k
1	Yudho	3673576	7270162	
2	Yova	4327843	7608566	
3	Budi	5438733	9673444	
4	Dwi	6989832	6982454	

↓
Tuples

- ◆ Entri dalam kolom bernilai atomik (tidak dapat dibagi)
- ◆ Entri dalam kolom berjenis sama
- ◆ Setiap baris adalah unik
- ◆ Urutan kolom (dari kiri ke kanan) bersifat bebas, tetapi urutan nilai harus mengikuti urutan kolom
- ◆ Urutan baris (dari atas ke bawah) bersifat bebas
- ◆ Setiap kolom mempunyai nama yang unik

- ◆ Harus *atomic* (bernilai tunggal)
- ◆ *Multivalued attribute* harus direpresentasikan pada relasi lain
- ◆ *Composite attribute* dinyatakan dalam simple attribute
- ◆ *Null value* bisa disebabkan karena tidak ada harga, atau tidak berguna

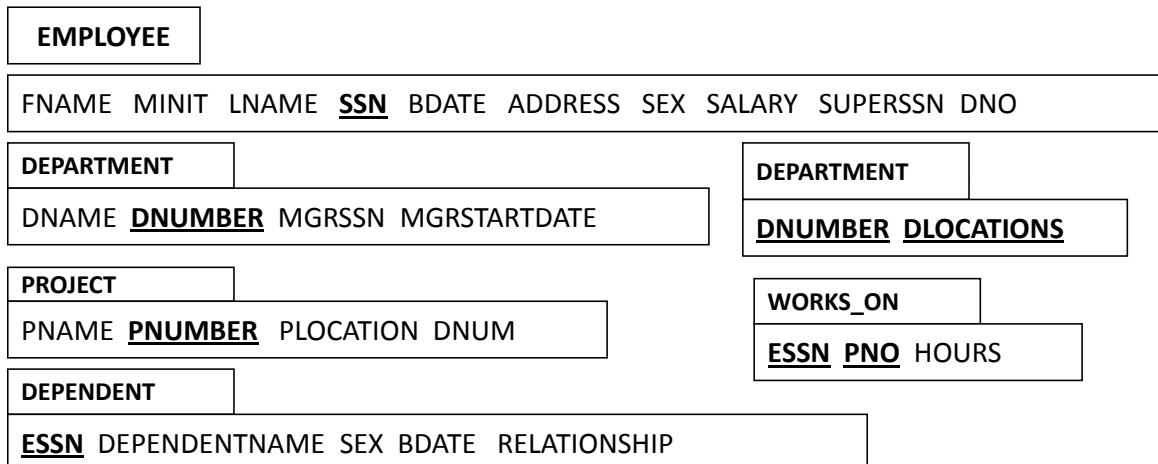
Key attribute pada Relasi

- ◆ Setiap baris dari suatu relasi mempunyai identifier yang unik, Satu atau gabungan kolom dapat menjadi identifier
- ◆ **Superkey:** kumpulan dari atribut yang bersifat unik
- ◆ **Key:**
 - ◆ *Superkey* yang minimal (tidak ada atribut yang dapat dihilangkan dari *superkey* yang membuat *superkey* tetap unik)
 - ◆ *Candidate key:* Sebuah relasi mungkin mempunyai lebih dari satu *key*. Masing-masing *key* disebut *candidate key*

- ◆ **Primary key:**
 - ◆ Candidate key yang dipilih untuk mengidentifikasi *tuple*/baris pada suatu relasi
 - ◆ Nilai primary key tidak boleh mengandung NULL dan harus unik.
- ◆ **Foreign key:**
 - ◆ Satu/beberapa atribut yang merujuk pada relasi lain yang merupakan *primary key attribute(s)* di relasi lain tsb dengan *domain* yang sama

Skema Model Relasional

- ◆ Dinyatakan dengan nama relasi diikuti dengan nama-nama kolomnya. Contoh:



Skema Model Relasional

- ◆ Contoh alternatif penulisan

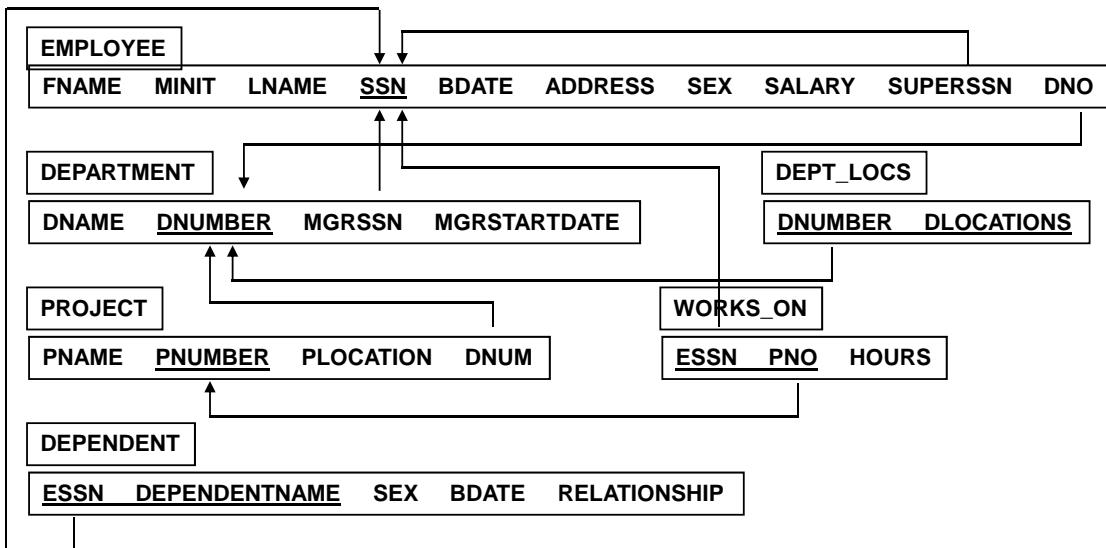
DEPARTMENT (DNAME, DNUMBER, MGRSSN,
MGRSTARTDATE)

dom(DNAME) = Department-names

dom(DNUMBERS) = Department-numbers

- ◆ Batasan diperlukan agar basis data selalu berada pada kondisi konsisten meski telah dilakukan pengubahan terhadap basis data tsb
- ◆ Batasan tsb disebut *integrity constraint*
- ◆ Macam-macam batasan:
 - ◆ *key constraint*: nilai *candidate key* harus unik
 - ◆ *entity integrity*: nilai *primary key* tidak boleh null

- ◆ *Referential integrity*: batasan untuk dua relasi yang saling berhubungan dimana bila tuple pada suatu relasi menunjuk tuple pada relasi lain, maka harus merujuk pada tuple yang benar-benar ada pada relasi lain tsb
- ◆ *Semantic integrity constraint*: batasan yang lebih umum, seperti gaji karyawan biasa tidak boleh lebih besar dari gaji manajer.
- ◆ Operasi pengolahan data pada relasi seperti *insert*, *delete*, dan *modify* harus menjaga agar batasan-batasan tsb tidak dilanggar



PERTEMUAN 5

Model Data Relational (Lanjut)

Pemetaan Diagram ER ke Model Relasional

-- 1 --

- ♦ Untuk setiap *entity type* E pada diagram ER, buat satu relasi R yang mengandung semua *simple attribute* dari E.
- ♦ Untuk *composite attribute*, masukkan komponen-komponen *simple attribute*-nya
- ♦ Pilih salah satu dari *key attribute* E sebagai *primary key* dari R
- ♦ Kalau *primary key*-nya bersifat *composite* maka kumpulan simple attribute yang membentuknya menjadi *primary key* juga

Pemetaan Diagram ER ke Model Relasional

-- 2 --

- ♦ Untuk setiap *weak entity type* W pada diagram ER dengan *owner entity type* E, buatlah suatu relasi R yang mengandung semua *simple attribute* W sebagai atribut dari R
- ♦ Masukkan *primary key* dari E sebagai *foreign key* dari R
- ♦ *Primary key* dari R adalah kombinasi dari *primary key* E dan *partial key* dari W
- ♦ Contoh: relasi DEPENDENT

Pemetaan Diagram ER ke Model Relasional

-- 3 --

- ◆ Untuk setiap *1:1 binary relationship type R* pada diagram ER, tentukan relasi S dan T yang berpartisipasi pada relationship type R
- ◆ Pilih salah satu relasi, misalkan S, masukkan primary key dari T sebagai foreign key S. Sebaiknya S adalah entity dengan total participation
- ◆ Masukkan semua atribut dari relationship type R sebagai atribut dari S. (Lihat relationship Manages, Department berpartisipasi total, muncul MGRSSN)

Pemetaan Diagram ER ke Model Relasional

- ◆ Alternatif: gabungkan kedua *entity type* dan *relationship* menjadi satu relasi.
- ◆ Hal ini dilakukan bila kedua *entity* yang berpartisipasi pada *relationship* mempunyai *total participation* dan *entity* tidak terlibat pada *relationship* lainnya.

Pemetaan Diagram ER ke Model Relasional

-- 4 --

- ♦ Untuk setiap 1:N *binary relationship type* R, tentukan relasi S pada sisi N dari relasi yang berpartisipasi pada *relationship* R.
- ♦ Masukkan primary key dari T sebagai *foreign key* dari S. Atribut dari R diambil sebagai atribut S
- ♦ (Lihat *relationship* WORKS_FOR dan SUPERVISION, muncul DNO dan SUPERSSN)

Pemetaan Diagram ER ke Model Relasional

-- 5 --

- ♦ Untuk M:N *binary relationship type* R, buat relasi S baru yang merepresentasikan R.
- ♦ Masukkan sebagai *foreign key* dari S, *primary key* dari *entity-entity* yang berpartisipasi pada *relationship*.
- ♦ Kombinasi dari *foreign key* tsb membentuk *primary key* dari S. Masukkan juga atribut dari *relationship* sebagai atribut S (Lihat relasi WORKS_ON)
- ♦ *Relationship* 1:1 dan 1:N dapat dipetakan dengan cara 5. Berguna bila *instance* hanya sedikit.

Pemetaan Diagram ER ke Model Relasional

-- 6 --

- ♦ Untuk setiap multivalued attribute A, buat suatu relasi R dengan memasukkan atribut A ditambah dengan primary key K dari relasi yang menyatakan entity/relationship type yang mempunyai A sebagai atribut.
- ♦ Primary key dari R ialah kombinasi dari A dan K (Lihat relasi DEPT_LOCS)

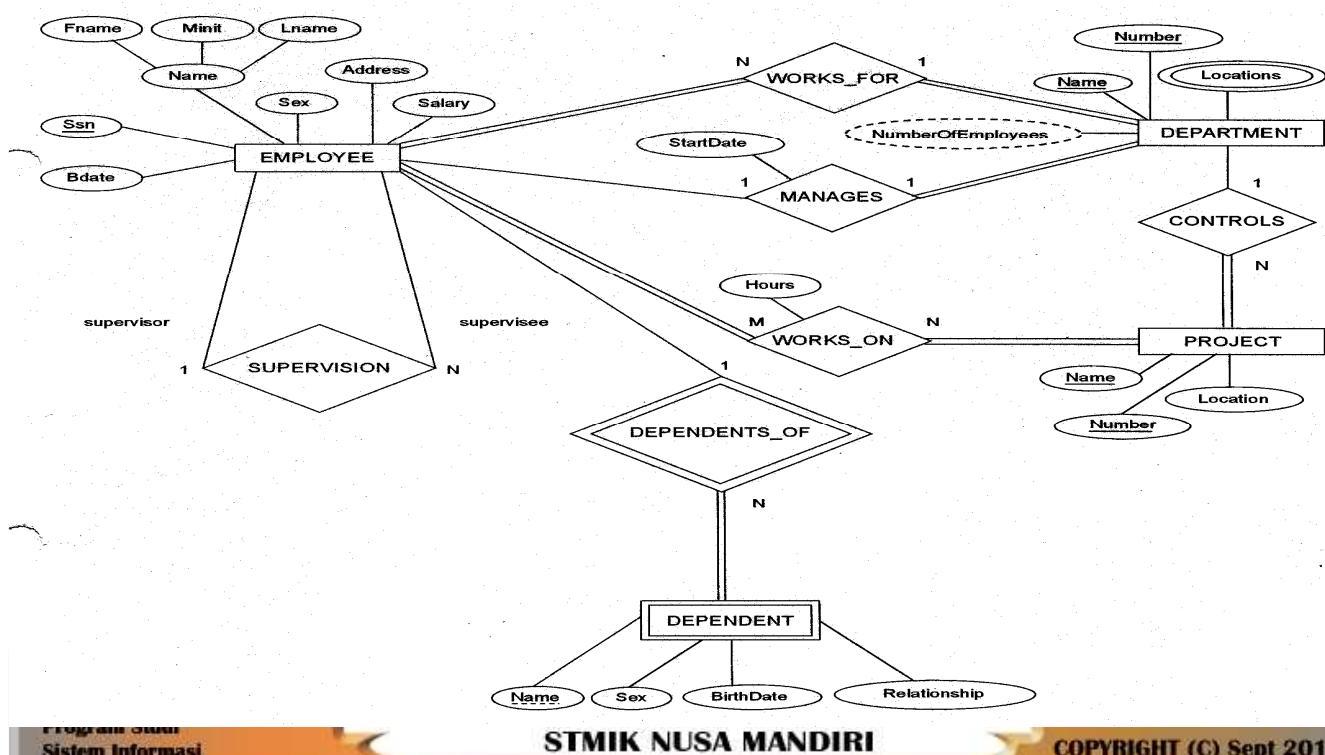
Pemetaan Diagram ER ke Model Relasional

-- 7 --

- ♦ Untuk tiap *n-ary relationship type* R dimana ($n > 2$), buat relasi baru S untuk menyatakan R.
- ♦ Masukkan *primary key* dari relasi-relasi yang berpartisipasi pada *relationship* sebagai *foreign key* dari S. *Simple attribute* dari R turut dimasukkan.
- ♦ *Primary key* dari S adalah kombinasi dari semua *foreign key* (Lihat contoh SUPPLIER)

Pemetaan Diagram ER ke Model Relasional

DIAGRAM MODEL ER

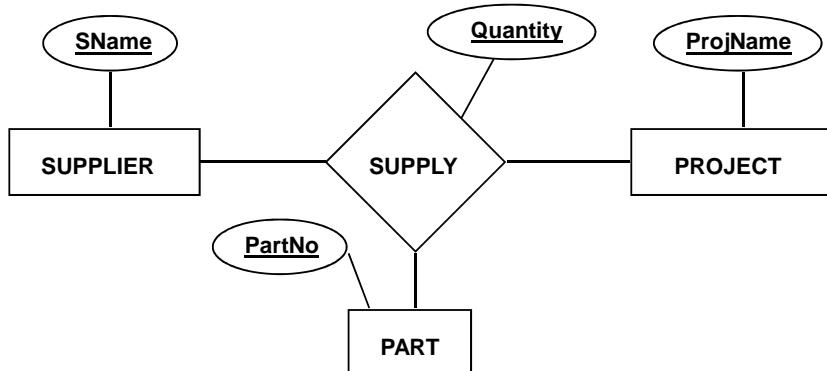


Pemetaan Diagram ER ke Model Relasional

SKEMA MODEL RELASIONAL

EMPLOYEE		f.k	f.k
FNAME	MINIT	LNAME	<u>SSN</u>
BDATE	ADDRESS	SEX	SALARY
			<u>SUPERSSN</u>
			DNO
p.k			
DEPARTMENT		f.k	
DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
p.k			
PROJECT			DEPT_LOCS
PNAME	<u>PNUMBER</u>	PLOCATION	DNUM
p.k		p.k	
DEPENDENT			WORKS_ON
ESSN	DEPENDENTNAME	SEX	BDATE
			RELATIONSHIP
p.k			
WORKS_ON			
	<u>ESSN</u>	PNO	HOURS
p.k			

Gambar Diagram ER



Gambar Model Relational

SUPPLIER
<u>SNAME</u> ...

PART
<u>PARTNO</u> ...

PROJECT
<u>PROJNAME</u> ...

SUPPLY			
SNAME	PROJNAME	PARTNO	QUANTITY

CRITICAL SUCCESS FACTOR PERANCANGAN BASISDATA

- ◆ Bekerja secara interaktif dengan pemakai(user)
- ◆ Memakai metodologi yang terstruktur
- ◆ Memakai pendekatan data-driven
- ◆ Memperhatikan struktur dan integritas dari model
- ◆ Memakai konsep visualisasi, normalisasi, validasi transaksi pada model
- ◆ Memakai diagram untuk menggambarkan model
- ◆ Melengkapi model dengan data-dictionary
- ◆ Mau mengulang langkah yang belum tepat.

- ◆ Ada beberapa model data yang ada dan yang paling populer saat ini adalah model data relasional.
- ◆ Produk-produk yang mengimplementasikan model data relasional juga telah banyak tersedia.
- ◆ Untuk membentuk model data relasional dari diagram ER telah tersedia langkah-langkah penuntunnya.

Perbedaan antara model Relational dengan Model ER

Model ER

- ◆ Entity type
- ◆ 1:1 atau 1:M relationship
- ◆ M:N relationship
- ◆ n-ary relationship
- ◆ simple attribute
- ◆ composite attribute
- ◆ multivalued attribute
- ◆ value set
- ◆ key attribute

Model Relasional

- ◆ entity relation
- ◆ foreign key atau realtionship relation
- ◆ relationship relation + 2 foreign key
- ◆ relationship relation + n foreign key
- ◆ attribute
- ◆ kumpulan dari simple attribute
- ◆ relation + foreign key
- ◆ domain
- ◆ primary key

PERTEMUAN 6

Normalisasi Database (Conoly-chap 14) (Ramakisman -chap 15)

Normalisasi adalah teknik desain database yang dimulai dengan memeriksa hubungan antar atribut.

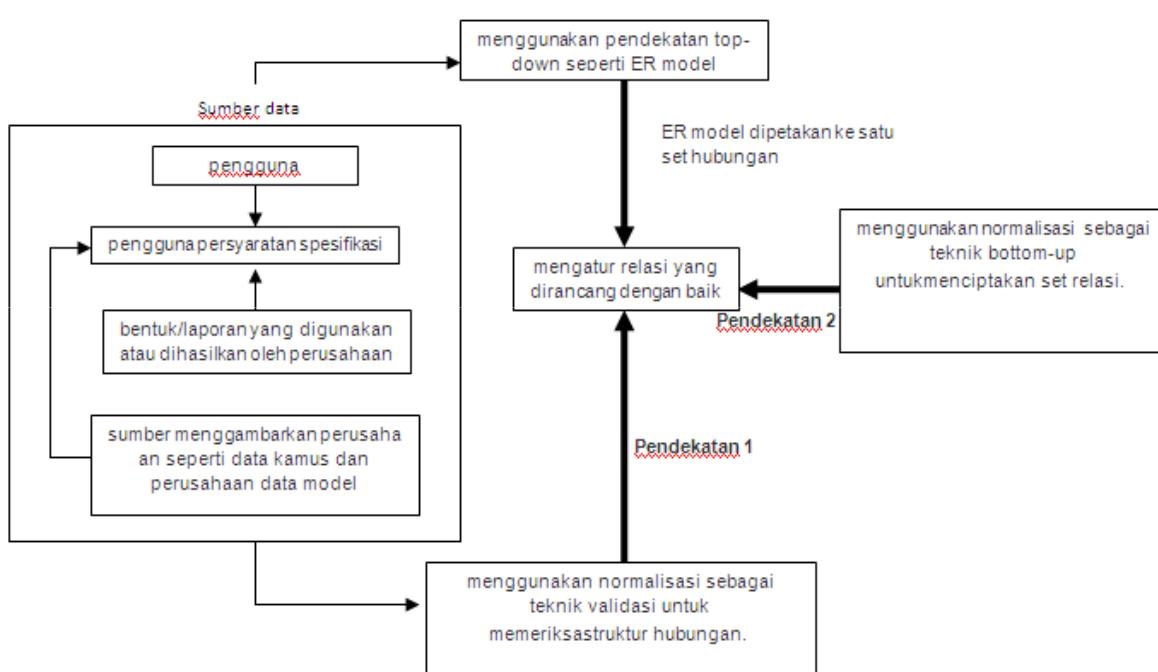
Dimana atribut menjelaskan dari data atau hubungan antara data. Normalisasi menggunakan serangkaian tes untuk membantu mengidentifikasi pengelompokan optimal untuk atribut-atribut yang akhirnya mengidentifikasi seperangkat hubungan sesuai dengan kebutuhan data perusahaan.

Bagaimana Normalisasi Mendesign Database

Dua pendekatan utama menggunakan normalisasi:

1. Pendekatan pertama menunjukkan bagaimana normalisasi dapat digunakan sebagai teknik bottom-up desain database mandiri
2. Pendekatan 2 menunjukkan bagaimana normalisasi dapat digunakan sebagai teknik validasi untuk memeriksa struktur hubungan, yang mungkin telah dibuat dengan menggunakan pendekatan top-down

Bagaimana normalisasi digunakan untuk desain database.



Pengantar Penyempurnaan Skema: Persoalan yang Ditimbulkan oleh Redundansi

- *Redundansi ruang penyimpanan*: beberapa data disimpan secara berulang
- *Update anomaly*: Jika satu copy data terulang tsb diubah, inkonsistensi data dpt terjadi kecuali kalau semua copy dari data tsb diubah dengan cara yang sama
- *Insertion anomaly*: Mungkin dpt terjadi kesulitan utk menyisipkan data tertentu kecuali kalau beberapa data tidak terkait lainnya juga ikut disisipkan
- *Deletion anomaly*: Mungkin dpt terjadi kesulitan utk menghapus data tertentu tanpa harus kehilangan beberapa data tidak terkait lainnya

Persoalan yang Ditimbulkan oleh Redundansi: Contoh

SSN	Name	Lot	Rating	Wages	Hours
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Asumsi nilai attribut wages ditentukan oleh nilai rating (utk satu nilai rating yang diberikan, hanya diperbolehkan terdapat satu nilai wages)

- *Redundansi ruang penyimpanan*: nilai rating 8 yang berkorespondensi dg wages 10 diulang tiga kali
- *Update anomaly*: Nilai wages (yg terkait dengan nilai rating) dlm baris pertama dpt diubah tanpa membuat perubahan yg sama pada baris kedua dan kelima

Persoalan yang Ditimbulkan oleh Redundansi: Contoh (cont'd)

SSN	Name	Lot	Rating	Wages	Hours
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Asumsi nilai attribut wages ditentukan oleh nilai rating (utk satu nilai rating yang diberikan, hanya diperbolehkan terdapat satu nilai wages)

- *Insertion anomaly:* Kesulitan utk menyisipkan employee baru kecuali nilai wage untuk rating dari employee tsb sudah diketahui
- *Deletion anomaly:* Jika semua baris yang terkait dg nilai rating tertentu dihapus (misalnya baris utk employee 'Smethurst' dan 'Guldu' dihapus), maka kita akan kehilangan informasi ketergantungan antara nilai rating dan nilai wages yang diasosiasikan dengan nilai rating tsb (yaitu rating = 5 dan wages = 7)

Penyebab Anomali

Mengapa anomali - anomali ini terjadi ?

- Karena menggabungkan dua tema (konsep entitas) dalam satu relasi. Ini mengakibatkan duplikasi – duplikasi sebagai akibat dari ketergantungan antar atribut yang tidak pada tempatnya.

Solusi : Normalisasi

Normalisasi

- *Normalisasi* adalah proses pembentukan struktur basis data sehingga sebagian besar *ambiguity* bisa dihilangkan.
- Tahap Normalisasi dimulai dari tahap paling ringan (1NF) hingga paling ketat (5NF)
- Biasanya hanya sampai pada tingkat 3NF atau BCNF karena sudah cukup memadai untuk menghasilkan tabel-tabel yang berkualitas baik.

Normalisasi

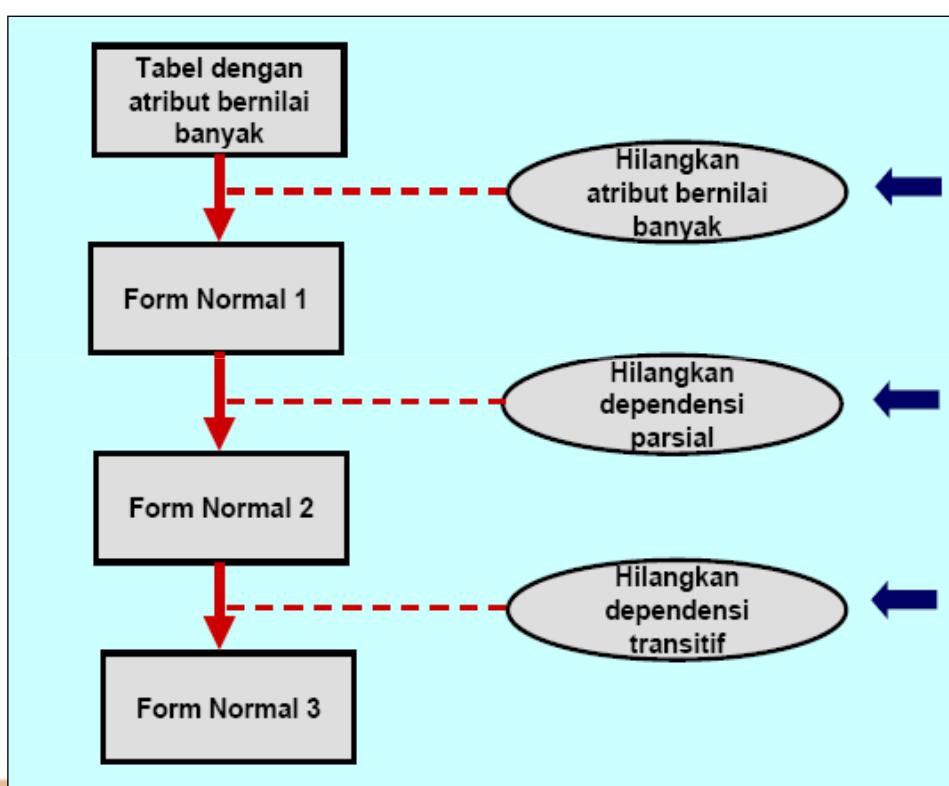
Sebuah tabel dikatakan baik (efisien) atau normal jika memenuhi 3 kriteria sbb:

1. Jika ada dekomposisi (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless-Join Decomposition*). Artinya, setelah tabel tersebut diuraikan / didekomposisi menjadi tabel-tabel baru, tabel-tabel baru tersebut bisa menghasilkan tabel semula dengan sama persis.
2. Terpeliharanya ketergantungan fungsional pada saat perubahan data (*Dependency Preservation*).
3. Tidak melanggar Boyce-Codd Normal Form (BCNF) (-akan dijelaskan kemudian-)

Normalisasi

Jika kriteria ketiga (BCNF) tidak dapat terpenuhi, maka paling tidak tabel tersebut tidak melanggar Bentuk Normal tahap ketiga (3rd Normal Form / 3NF).

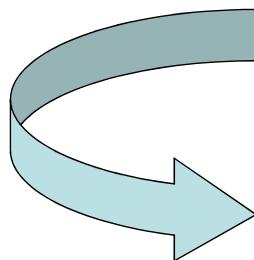
Langkah – Langkah Normalisasi



Tabel Universal

Tabel Universal (*Universal / Star Table*) → sebuah tabel yang merangkum semua kelompok data yang saling berhubungan, bukan merupakan tabel yang baik.

Misalnya:



Tabel Universal

	nrp	mhs.nama	mhs.alamat	kodekul	namakul	skls	kodesem	nihiluruf	dsn.nama	dsn.alamat
▶	11020001	Abdullah Machrus	Jl. Sinoman 1/11 Mojokerto	SP	Software Perkantoran	21		A	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020002	Achmad Fajril	Jl. Panglima Sudirman XII / 30	SP	Software Perkantoran	21		A	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020003	Achmad Ridho	Geluran RT 13 / 03 Sepanjang S	SP	Software Perkantoran	21		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020004	Adi Christanto	Jl. Wonorejo IV / 45 Surabaya	SP	Software Perkantoran	21		AB	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020005	Aloysius Rendy	Pucangan VII / 9 Surabaya	SP	Software Perkantoran	21		D	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020006	Anita Rachmawati	Perum Canda Bhirawa Asri N - 1	SP	Software Perkantoran	21		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020007	Arif Fachrudin	Jl. Gubernur Suryo No.15	SP	Software Perkantoran	21		E	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby
	11020008	Arohman Agung	Kupang Gunung Timur IV / 24 A	SP	Software Perkantoran	21		C	Imam Kuswardayan	Jl. Teknik Komputer 18 Sby

Functional Dependency

- Notasi: $A \rightarrow B$

A dan B adalah atribut dari sebuah tabel. Berarti secara fungsional A menentukan B atau B tergantung pada A, jika dan hanya jika ada 2 baris data dengan nilai A yang sama, maka nilai B juga sama

- Notasi: $A \not\rightarrow B$ atau $A \times \rightarrow B$

Adalah kebalikan dari notasi sebelumnya.

Functional Dependency

Contoh tabel nilai

Namakul	Nrp	namamhs	NiHuruf
Struktur Data	980001	Ali Akbar	A
Struktur Data	980004	Indah Susanti	B
Basis Data	980001	Ali Akbar	
Basis Data	980002	Budi Haryanto	
Basis Data	980004	Indah Susanti	
Bahasa Indonesia	980001	Ali Akbar	B
Matematika I	980002	Budi Haryanto	C

Functional Dependency

Functional Dependency dari tabel nilai

- **Nrp → namaMhs**

Karena untuk setiap nilai nrp yang sama, maka nilai namaMhs juga sama.

- **{Namakul, nrp} → NiHuruf**

Karena attribut Nihuruf tergantung pada Namakul dan nrp secara bersama-sama. Dalam arti lain untuk Namakul dan nrp yang sama, maka NiHuruf juga sama, karena Namakul dan nrp merupakan key (bersifat unik).

- **NamaKul ↗ nrp**
- **Nrp ↗ NiHuruf**

Bentuk-bentuk Normal

1. Bentuk Normal Tahap Pertama (1st Normal Form / 1NF)
2. Bentuk Normal Tahap Kedua (2nd Normal Form / 2NF)
3. Bentuk Normal Tahap (3rd Normal Form / 3NF)
4. Boyce-Codd Normal Form (BCNF)
5. Bentuk Normal Tahap (4th Normal Form / 4NF)
6. Bentuk Normal Tahap (5th Normal Form / 5NF)

Bentuk Normal Tahap Pertama (1st Normal Form / 1NF)

- Bentuk normal 1NF terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (*multivalued attribute*), atribut composite atau kombinasinya dalam domain data yang sama.
- Setiap atribut dalam tabel tersebut harus bernilai *atomic* (tidak dapat dibagi-bagi lagi)

Contoh 1

Misal data mahasiswa sbb:

Nrp	nama	Hobi
12020001	Heri Susanto	Sepakbola, membaca komik, berenang
12020013	Siti Zulaiha	Memasak,mrogram komputer
12020015	Dini Susanti	Menjahit,membuat roti

Atau:

Nrp	nama	hobi1	hobi2	Hobi3
12020001	Heri Susanto	Sepak Bola	Membaca komik	berenang
12020013	Siti Zulaiha	Memasak	mrogram komputer	
12020015	Dini Susanti	Menjahit	membuat kue	

Tabel-tabel di atas tidak memenuhi syarat 1NF

Contoh 1

Didekomposisi menjadi:

- Tabel Mahasiswa

Nrp	Nama
12020001	Heri Susanto
12020013	Siti Zulaiha
12020015	Dini Susanti

- Tabel Hobi

Nrp	Hobi
12020001	Sepakbola
12020001	membaca komik
12020001	Berenang
12020013	Memasak
12020013	rogram komputer
12020015	Menjahit
12020015	membuat roti

Contoh 2 (composite)

JadwalKuliah

Kodekul	NamaKul	Dosen	Kelas	Jadwal

- Dimana nilai pada atribut jadwal berisi gabungan antara Hari dan Jam.
- Jika asumsi hari dan jam memegang peranan penting dalam sistem basis data, maka atribut Jadwal perlu dipisah sehingga menjadi JadwalHari dan JadwalJam sbb:

JadwalKuliah

Kodekul	NamaKul	Dosen	Kelas	JadwalHari	JadwalJam

Bentuk Normal Tahap Kedua (2nd Normal Form)

- Bentuk normal 2NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk 1NF, dan semua atribut selain primary key, secara utuh memiliki Functional Dependency pada primary key
- Sebuah tabel tidak memenuhi 2NF, jika ada atribut yang ketergantungannya (Functional Dependency) hanya bersifat parsial saja (hanya tergantung pada sebagian dari primary key)
- Jika terdapat atribut yang tidak memiliki ketergantungan terhadap primary key, maka atribut tersebut harus dipindah atau dihilangkan

Contoh

Tabel berikut memenuhi 1NF tapi tidak termasuk 2NF:

Mhs_nrp	mhs_nama	mhs_alamat	mk_kode	mk_nama	mk_sks	nihiluruf
---------	----------	------------	---------	---------	--------	-----------

- Tidak memenuhi 2NF, karena $\{Mhs_nrp, mk_kode\}$ yang dianggap sebagai primary key sedangkan:
 - $\{Mhs_nrp, mk_kode\} \not\rightarrow mhs_nama$
 - $\{Mhs_nrp, mk_kode\} \not\rightarrow mhs_alamat$
 - $\{Mhs_nrp, mk_kode\} \not\rightarrow mk_nama$
 - $\{Mhs_nrp, mk_kode\} \not\rightarrow mk_sks$
 - $\{Mhs_nrp, mk_kode\} \rightarrow nihiluruf$
- Tabel di atas perlu didekomposisi menjadi beberapa tabel yang memenuhi syarat 2NF

Contoh

Functional dependencynya sbb:

$$\{Mhs_nrp, mk_kode\} \rightarrow nihuruf \quad (fd1)$$

$$Mhs_nrp \rightarrow \{mhs_nama, mhs_alamat\} \quad (fd2)$$

$$Mk_kode \rightarrow \{mk_nama, mk_skls\} \quad (fd3)$$

fd1 (mhs_nrp, mk_kode, nihuruf) → Tabel Nilai

fd2 (Mhs_nrp, mhs_nama, mhs_alamat) → Tabel Mahasiswa

fd3 (mk_kode, mk_nama, mk_skls) → Tabel MataKuliah

Bentuk Normal Tahap Ketiga (3rd Normal Form /3NF)

- Bentuk normal 3NF terpenuhi jika telah memenuhi bentuk 2NF, dan jika **tidak ada** atribut *non primary key* yang memiliki ketergantungan terhadap atribut *non primary key* yang lainnya.
- Untuk setiap Functional Dependency dengan notasi $X \rightarrow A$, maka:
 - X harus menjadi superkey pada tabel tsb.
 - Atau A merupakan bagian dari primary key pada tabel tsb.

Contoh

Tabel berikut memenuhi 2NF, tapi tidak memenuhi 3NF:

Mahasiswa

Nrp	Nama	Alm_Jalan	Alm_Kota	Alm_Provinsi	Alm_Kodepos
-----	------	-----------	----------	--------------	-------------

- karena masih terdapat atribut *non primary key* (yakni **alm_kota** dan **alm_Provinsi**) yang memiliki ketergantungan terhadap atribut *non primary key* yang lain (yakni **alm_kodepos**):

$$\text{alm_kodepos} \rightarrow \{\text{alm_Provinsi}, \text{alm_kota}\}$$

- Sehingga tabel tersebut perlu didekomposisi menjadi:

Mahasiswa (Nrp, nama, alm_jalan, alm_kodepos)

Kodepos (alm_kodepos, alm_provinsi, alm_kota)

Boyce-Codd Normal Form (BCNF)

- Bentuk BCNF terpenuhi dalam sebuah tabel, jika untuk setiap *functional dependency* terhadap setiap atribut atau gabungan atribut dalam bentuk: $X \rightarrow Y$ maka X adalah **super key**
- tabel tersebut harus di-dekomposisi berdasarkan *functional dependency* yang ada, sehingga X menjadi **super key** dari tabel-tabel hasil dekomposisi
- Setiap tabel dalam BCNF merupakan 3NF. Akan tetapi setiap 3NF belum tentu termasuk BCNF . Perbedaannya, untuk *functional dependency* $X \rightarrow A$, BCNF tidak membolehkan A sebagai bagian dari primary key.



Bentuk Normal Tahap Keempat (4th Normal Form /4NF)

- Bentuk normal 4NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk BCNF, dan tabel tersebut tidak boleh memiliki lebih dari sebuah *multivalued attribute*
- Untuk setiap *multivalued dependencies* (MVD) juga harus merupakan *functional dependencies*



Contoh

Misal, tabel berikut tidak memenuhi 4NF:

Employee	Project	Skill
Jim	11	Program
Mary	5	Design
Mary	NULL	Analysis

Setiap employee dapat bekerja di lebih dari project dan dapat memiliki lebih dari satu skill. Untuk kasus seperti ini tabel tersebut harus di-dekomposisi menjadi:

(Employee, Project)
(Employee, Skill)

Bentuk Normal Tahap Keempat (5th Normal Form /5NF)

- Bentuk normal 5NF terpenuhi jika tidak dapat memiliki sebuah *lossless decomposition* menjadi tabel-tabel yg lebih kecil.
- Jika 4 bentuk normal sebelumnya dibentuk berdasarkan *functional dependency*, 5NF dibentuk berdasarkan konsep *join dependence*. Yakni apabila sebuah tabel telah di-dekomposisi menjadi tabel-tabel lebih kecil, harus bisa digabungkan lagi (join) untuk membentuk tabel semula

PERTEMUAN 9

Penyempurnaan Skema dan Bentuk-bentuk Normal

Pokok Bahasan

- Persoalan-persoalan apa yang dapat ditimbulkan oleh adanya redundansi penyimpanan informasi?
- Apa yang dimaksud dengan *functional dependencies*?
- Apa yang dimaksud dengan bentuk-bentuk normal (*normal forms*) dan apa tujuannya?
- Apa manfaat dari BCNF dan 3NF?
- Apa pertimbangan dalam mendekomposisi relasi-relasi menjadi bentuk-bentuk normal?
- Dimana proses normalisasi dapat digunakan dalam proses desain basis data?
- Adakah bentuk kebergantungan (*dependency*) umum yang lebih bermanfaat dalam desain basis data?

Pengantar Penyempurnaan Skema: Persoalan yang Ditimbulkan oleh Redundansi

- *Redundansi ruang penyimpanan*: beberapa data disimpan secara berulang
- *Update anomaly*: Jika satu copy data terulang tsb diubah, inkonsistensi data dpt terjadi kecuali kalau semua copy dari data tsb diubah dengan cara yang sama
- *Insertion anomaly*: Mungkin dpt terjadi kesulitan utk menyisipkan data tertentu kecuali kalau beberapa data tidak terkait lainnya juga ikut disisipkan
- *Deletion anomaly*: Mungkin dpt terjadi kesulitan utk menghapus data tertentu tanpa harus kehilangan beberapa data tidak terkait lainnya

Persoalan yang Ditimbulkan oleh Redundansi: Contoh

SSN	Name	Lot	Rating	Wages	Hours
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Asumsi: nilai attribut wages ditentukan oleh nilai rating (utk satu nilai rating yang diberikan, hanya diperbolehkan terdapat satu nilai wages)

- *Redundansi ruang penyimpanan:* nilai rating 8 yang berkorespondensi dg wages 10 diulang tiga kali
- *Update anomaly:* Nilai wages (yg terkait dengan nilai rating) dlm baris pertama dpt diubah tanpa membuat perubahan yg sama pada baris kedua dan kelima

Persoalan yang Ditimbulkan oleh Redundansi: Contoh

SSN	Name	Lot	Rating	Wages	Hours
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Asumsi: nilai attribut wages ditentukan oleh nilai rating (utk satu nilai rating yang diberikan, hanya diperbolehkan terdapat satu nilai wages)

- *Insertion anomaly:* Kesulitan utk menyisipkan employee baru kecuali nilai wage untuk rating dari employee tsb sudah diketahui
- *Deletion anomaly:* Jika semua baris yang terkait dg nilai rating tertentu dihapus (misalnya baris utk employee 'Smethurst' dan 'Guldu' dihapus), maka kita akan kehilangan informasi ketergantungan antara nilai rating dan nilai wages yang diasosiasikan dengan nilai rating tsb (yaitu rating = 5 dan wages = 7)

Persoalan yang Ditimbulkan oleh Redundansi: Null Values

- Untuk kasus-kasus khusus, adanya nilai-nilai *null* yang berlebihan dalam suatu relasi dpt menimbulkan pemborosan penggunaan ruang penyimpanan.
- Hal ini terutama dpt terjadi pada suatu relasi dengan jumlah attribut yang besar dan jumlah baris yang juga besar, sehingga untuk kasus tertentu dapat terjadi banyak nilai-nilai kolom yang tidak memenuhi (*not applicable*) untuk sejumlah baris dalam relasi harus dibiarkan bernilai *null*.
- Sebagai contoh, utk relasi “Hourly Employees”, misalkan ditambah satu kolom baru (*OfficeLocCode*) utk mencatat kode lokasi kantor dari para pemimpin perusahaan. Jika misalnya terdapat ribuan employee, dan hanya ada sekitar 10% pemimpin, maka sebagian besar (90%) nilai kolom tersebut akan terisi dengan nilai *null* (pemborosan ruang penyimpanan).

Pengantar Penyempurnaan Skema: Dekomposisi Skema Relasi

- Proses Dekomposisi sebuah skema relasi *R* berupa penggantian skema relasi menjadi dua (atau lebih) skema-skema baru yang masing-masing berisikan subset dari attribut-attribut relasi *R* dan kesemuanya memuat semua attribut yang ada dalam relasi *R*.
 - Proses dekomposisi dilakukan dengan menggunakan konsep ketergantungan fungsional (functional dependencies)
- Contoh: skema relasi “Hourly_Employees” dpt didekomposisi menjadi:
 - Hourly_Emps2 (*ssn*, *name*, *lot*, *rating*, *hours*)
 - Wages (*rating*, *wages*)

Hourly_Emps2

Wages	
R	W
8	10
5	7

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40

Dekomposisi Skema Relasi: Beberapa Persoalan Terkait

- Dekomposisi terhadap suatu skema relasi harus digunakan dengan penuh pertimbangan. Dua pertanyaan yang harus selalu dipertimbangkan:
 1. Adakah alasan untuk mendekomposisi suatu relasi ?
 2. Persoalan-persoalan apa saja (jika ada) yang akan diakibatkan oleh dekomposisi ?
- Jawaban thdp pertanyaan pertama dpt dibantu dengan bentuk-bentuk normal (*Normal Forms/NF*) terhadap relasi yang akan didekomposisi. Utk ini jika suatu skema relasi berada dlm salah satu NF, maka beberapa persoalan yang terkait dengan dekomposisi tidak akan muncul
- Untuk jawaban thdp pertanyaan kedua, dua sifat penting dari dekomposisi harus dipertimbangkan:
 - Sifat *lossless-join* yang memungkinkan untuk membentuk kembali (recover) nilai-nilai relasi yang didekomposisi dari relasi-relasi hasil dekomposisi
 - Sifat *dependency-preservation* yang memungkinkan untuk memaksa agar constraints yang berlaku pada relasi asal tetap berlaku pada sejumlah relasi-relasi yang lebih kecil

Functional Dependencies (FDs)

- Suatu functional dependency $X \rightarrow Y$ dikatakan berlaku pada relasi R jika, utk setiap nilai r dari R yang diperbolehkan, berlaku keadaan:
 - $t_1 \in r, t_2 \in r, \pi_X(t_1) = \pi_X(t_2)$ mengimplikasikan $\pi_Y(t_1) = \pi_Y(t_2)$
 - yaitu, jika diberikan dua tuples dalam r , jika nilai proyeksi X pada kedua tuples sama, maka nilai proyeksi Y pada kedua tuples juga sama. (X dan Y adalah sets dari attributes pada relasi yang sama.)
- Sebuah FD adalah pernyataan yang berlaku pada semua relasi-relasi yang dimungkinkan.
 - Harus diidentifikasi berdasarkan semantik dari aplikasi
 - Jika diberikan beberapa nilai r_1 dari R yang mungkin, kita dpt melakukan pengecekan apakah nilai tersebut melanggar beberapa FD f , tetapi kita tidak dapat mengatakan bahwa f berlaku pada R!
- Jika K adalah sebuah *candidate key* untuk R, maka berarti bahwa $K \rightarrow R$
 - Tetapi, $K \rightarrow R$ tidak mengharuskan K terdiri dari satu set attribut yang *minimal* !

Contoh: Constraints pada Entity Set

- Perhatikan relasi Hourly_Emps berikut:
 - Hourly_Emps (ssn, name, lot, rating, *hrly_wages*, hrs_worked)
- Notasi: Utk penyederhaan penulisan, skema relasi tsb akan dinotasikan dengan menggabungkan singkatan dari attribut-attributnya: SNLRWH
 - Notasi ini menyatakan satu set attributes {S,N,L,R,W,H}.
 - Dalam beberapa kasus, nama sebuah relasi akan digunakan untuk mengacu ke semua attribut dari relasi tersebut. (contoh, Hourly_Emps untuk SNLRWH)
- Beberapa FD yang berlaku pada Hourly_Emps:
 - *ssn* adalah sebuah key: $S \rightarrow \text{SNLRWH}$
 - *rating* menentukan *hrly_wages*: $R \rightarrow W$

Contoh (Lanjutan)

- Beberapa persoalan akibat $R \rightarrow W$:
 - Update anomaly: Dapatkah *W* diubah hanya pada tuple pertama dari SNLRWH ?
 - Insertion anomaly: Bgm jika diinginkan utk menyisipkan seorang *employee* tetapi *hourly wage* utk *rating* yang bersangkutan tidak diketahui ?
 - Deletion anomaly: Jika semua *employee* dengan *rating* 5 dihapus, maka informasi mengenai *hourly wage* utk *rating* 5 juga akan ikut terhapus

SSN	Name	Lot	Rating	Wages	Hours
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

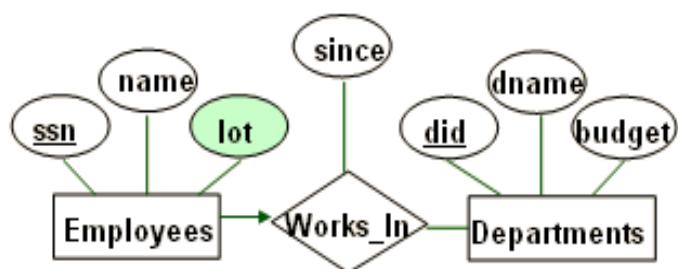
Wages	
R	W
8	10
5	7

Hourly_Emps2					
S	N	L	R	H	
123-22-3666	Attishoo	48	8	40	
231-31-5368	Smiley	22	8	30	
131-24-3650	Smethurst	35	5	30	
434-26-3751	Guldu	35	5	32	
612-67-4134	Madayan	35	8	40	

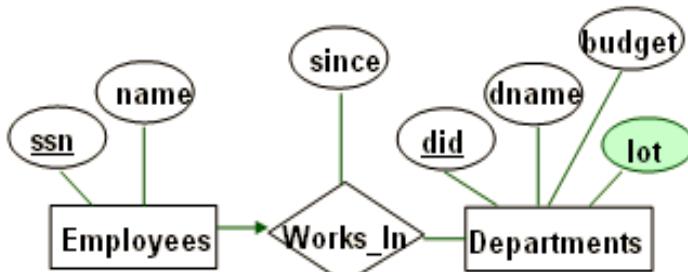
Penyempurnaan ER Diagram

- Pemetaan diagram pertama:
Workers(S,N,L,D,S)
Departments(D,M,B)
 - Lots diasosiasikan dengan relasi workers.
- Jika diasumsikan bhw semua workers dlm sebuah dept ditentukan sebuah lot yang sama, maka D → L (Redundansi)
- Redundansi dpt diatasi dg:
Workers2(S,N,D,S)
Dept_Lots(D,L)
- Dapat disempurnakan (*fine-tune*) menjadi:
Workers2(S,N,D,S)
Departments(D,M,B,L)

Sebelum (Workers):



Sesudah (Workers2):



Alasan Mengenai FD

- Jika diberikan satu set FDs, kita dapat menurunkan (*infer*) tambahan FDs: $ssn \rightarrow did$, $did \rightarrow lot$ mengimplikasikan $ssn \rightarrow lot$
- Sebuah FD f dikatakan dapat diimplikasikan oleh (*implied by*) satu set FDs F jika f berlaku bilamana semua FDs dalam F berlaku.
 - F^+ (*closure of F*) adalah set dari semua FDs yang diimplikasikan oleh F .
- Aksioma Armstrong (X, Y, Z adalah sets dari attributes):
 - Reflexivity: Jika $X \supseteq Y$, maka $X \rightarrow Y$
 - Augmentation: Jika $X \rightarrow Y$, maka $XZ \rightarrow YZ$ utk sembarang Z
 - Transitivity: Jika $X \rightarrow Y$ dan $Y \rightarrow Z$, maka $X \rightarrow Z$
- Aksioma di atas merupakan aturan-aturan penyimpulan (*inference rules*) untuk FDs yang *logis* (*sounds*) dan *lengkap* (*complete*) !
- Dua aturan tambahan yang menyertai Aksioma Armstrong:
 - Union: Jika $X \rightarrow Y$ dan $X \rightarrow Z$, maka $X \rightarrow YZ$
 - Decomposition: Jika $X \rightarrow YZ$, maka $X \rightarrow Y$ dan $X \rightarrow Z$

Alasan Mengenai FD (Lanjutan)

- Suatu FD disebut *trivial* jika sisi kanan dari FD hanya terdiri dari attribut yang juga muncul di sisi kiri dari FD (akibat rumus *reflexivity*). Selain trivial FDs, selebihnya disebut *nontrivial* FDs.
 - Dengan menggunakan rumus reflexivity, semua trivial dependencies dapat diturunkan.
- Contoh: Contracts(*cid,sid,jid,did,pid,qty,value*), dan:
 - C adalah key: $C \rightarrow CSJDPQV$
 - Project membeli setiap part menggunakan contract tunggal: $JP \rightarrow C$
 - Dept membeli paling banyak satu part dari sebuah supplier: $SD \rightarrow P$
- Nontrivial FDs yang dapat diperoleh dari relasi ‘Contracts’:
 - $JP \rightarrow C$, $C \rightarrow CSJDPQV$ mengimplikasikan $JP \rightarrow CSJDPQV$
 - $SD \rightarrow P$ mengimplikasikan $SDJ \rightarrow JP$
 - $SDJ \rightarrow JP$, $JP \rightarrow CSJDPQV$ mengimplikasikan $SDJ \rightarrow CSJDPQV$

Alasan Mengenai FD: Attribute Closure

- Hanya untuk mengecek apakah sebuah dependensi, misalnya $X \rightarrow Y$ terdapat dlm closure dari satu set FDs F , kita dapat melakukannya secara efisien tanpa harus menghitung F^+ .
 - Hitung attribute closure X^+ dg mengacu pada F , yaitu hitung satu attributes A sehingga $X \rightarrow A$ dpt diturunkan menggunakan Aksioma Armstrong (Algoritma utk menghitung attribute closure X^+ dari set attribut X dg mengacu pada satu set FDs F):


```
closure = X;
REPEAT
{
  IF there is an FD U → V in F such that U ⊆ closure THEN
    set closure = closure ∪ V
} UNTIL there is no change;
```
 - Kemudian cek apakah Y ada dalam X^+
- Latihan: Apakah $F = \{A \rightarrow B, B \rightarrow C, CD \rightarrow E\}$ mengimplikasikan $A \rightarrow E$?
- Algoritma di atas dpt dimodifikasi utk memperoleh “keys” dari suatu skema relasi dengan cara memulai dengan set X yang terdiri dari satu attribut tunggal dan berhenti begitu closure berisikan semua attribut dari skema relasi.

Bentuk-Bentuk Normal (Normal Forms)

- Normal Forms (NF) digunakan utk membantu kita utk memutuskan apakah suatu skema relasi sudah merupakan hasil desain yang "baik" atau masih perlu didekomposisi menjadi relasi-relasi yang lebih kecil.
- Jika suatu relasi skema sudah berada dalam salah satu NF, berarti bhw beberapa jenis persoalan redundansi dapat dihindari/diminimalkan.
- NF yang didasarkan pada FDs: 1NF, 2NF, 3NF dan Boyce-Codd NF (BCNF):
 - Setiap relasi dlm BCNF juga berada dlm 3 NF
 - Setiap relasi dlm 3 NF juga berada dlm 2 NF, dan
 - Setiap relasi dlm 2 NF juga berada dlm 1NF

Bentuk-Bentuk Normal (Normal Forms)

- Setiap relasi yang berada dlm 1NF berlaku constraint bhw setiap field hanya berisikan nilai-nilai atomic (tidak boleh berisikan lists atau sets).
 - Dlm perkuliahan, constraint ini dianggap berlaku sebelum dilakukan proses normalisasi
- Oleh karena 2NF dibuat atas dasar sejarah perkembangan database (dari network model ke relational model), maka pembahasan hanya ditekankan pada proses pembentukan 3NF dan BCNF yang merupakan langkah penting dalam proses desain database.

Bentuk-Bentuk Normal (Normal Forms)

- Peran FD dalam mendeteksi redundansi: Perhatikan sebuah relasi R dengan 3 attributes ABC.
 - Jika tidak ada FD yang berlaku pada relasi R, maka dapat dipastikan tidak akan terdapat persoalan redundansi.
 - Namun, jika dimisalkan berlaku $A \rightarrow B$, maka jika terdapat beberapa tuples yang mempunyai nilai A yang sama maka baris-baris tersebut juga harus mempunyai nilai B yang sama. Untuk ini, potensi terjadinya redundansi dapat diperkirakan dengan menggunakan informasi FDs

Boyce-Codd Normal Form (BCNF)

- Relasi R dg FDs F dikatakan berada dalam BCNF jika, utk semua FD $X \rightarrow A$ dalam F , salah satu dari pernyataan berikut harus berlaku:
 - $A \in X$ (disebut *trivial* FD), atau
 - X adalah key dari R.
- Dengan kata lain, R dikatakan berada dalam BCNF jika non-trivial FDs yang berlaku pada R hanya berupa key constraints.
 - Tidak ada redundansi yang dpt diprediksi hanya dengan menggunakan FDs saja
 - Jika terdapat dua tuples yang mempunyai nilai X yang sama, maka kita tidak dapat menyimpulkan nilai A dalam satu tuple dari nilai A dalam tuple lainnya
 - Namun, jika relasi contoh berada dalam BCNF, maka kedua tuples harus identik (karena X adalah sebuah key).

X	Y	A
x	y1	a
x	y2	?

Third Normal Form (3NF)

- Relasi R dg FDs F dikatakan berada dlm 3NF jika, untuk semua FD $X \rightarrow A$ dalam F , salah satu dari pernyataan berikut harus berlaku:
 - $A \in X$ (disebut *trivial FD*), atau
 - X adalah key dari R , atau
 - A adalah bagian dari beberapa key dari R (A adalah *prime attribute*)
- *Minimality* dari key dalam kondisi ketiga di atas menjadi sangat penting !
- Jika R berada dlm BCNF, sudah tentu juga berada dlm 3NF
- Jika R berada dlm 3NF, beberapa redundansi masih mungkin terjadi.
 - Bentuk 3NF dapat dipakai sebagai bentuk yang kompromistik dan digunakan bilamana BCNF tidak dapat diupayakan (misalnya karena tidak ada dekomposisi yang “baik”, atau karena alasan pertimbangan kinerja dari database)

Apa yang Dapat Dicapai oleh 3NF?

- Jika depedensi $X \rightarrow A$ menyebabkan pelanggaran dari 3NF, maka salah satu kasus di bawah ini akan terjadi:
 - X adalah subset dari beberapa key K (*partial dependency*)
 - Pasangan nilai (X, A) yang sama akan tersimpan secara redundan
 - X bukan subset dari sembarang key K (*transitive dependency*)
 - Terdapat mata rantai FDs $K \rightarrow X \rightarrow A$, yang berarti bhw kita tdk dpt mengasosiasikan sebuah nilai X dengan sebuah nilai K kecuali kalau kita juga mengasosiasikan sebuah nilai A dengan sebuah nilai X
- Namun demikian, walaupun seandainya relasi berada dalam 3NF, persoalan-persoalan berikut masih dpt terjadi:
 - Contoh: relasi Reserves SBDC (C =Credit Card ID), $S \rightarrow C$, $C \rightarrow S$ berada dalam 3NF, tetapi utk setiap reservasi dari sailor S , pasangan nilai (S, C) yang sama akan tersimpan dalam database.
 - Dengan demikian, 3NF memang merupakan bentuk normal yang relatif kompromistik terhadap BCNF.

Proses Dekomposisi dari sebuah Skema Relasi

- Asumsikan relasi R terdiri dari attributes $A_1 \dots A_n$.
Proses dekomposisi dari R meliputi penggantian R oleh dua atau lebih relasi sehingga :
 - Setiap skema relasi yang baru terdiri dari subset attribut dari R (dan tidak satupun attribut yang tidak muncul dalam R), dan
 - Setiap attribut dari R muncul sebagai sebuah attribut dari salah satu relasi-relasi yang baru
- Secara intuitif, pendekomposisian R berarti bahwa kita akan menyimpan nilai-nilai dari skema-skema relasi yang dihasilkan oleh proses dekomposisi, bukan nilai-nilai dari relasi R
- Contoh, relasi SNLRWH dapat didekomposisi menjadi SNLRH dan RW (lihat slide berikutnya).

Contoh Dekomposisi

- Proses dekomposisi sebaiknya digunakan hanya bilamana diperlukan.
 - SNLRWH mempunyai FDs $S \rightarrow SNLRWH$ dan $R \rightarrow W$
 - FD kedua menimbulkan pelanggaran 3NF; nilai-nilai W secara berulang diasosiasikan dg nilai-nilai R. Cara yang termudah utk memperbaiki ini adalah menciptakan relasi baru RW utk menyimpan asosiasi-asosiasi tersebut, dan untuk menghapus W dari skema utama, yaitu:
 - Kita dekomposisi SNLRWH menjadi SNLRH dan RW
- Informasi yang akan disimpan terdiri dari SNLRWH tuples. Jika kita hanya menyimpan proyeksi dari tuples ini pada SNLRH dan RW, adakah persoalan-persoalan potensial lain yang perlu dipertimbangkan? (lihat slide berikutnya)

Persoalan-persoalan yang Dapat Ditimbulkan oleh Dekomposisi

- Terdapat 3 persoalan potensial yang perlu diperhatikan:
 - Beberapa queries menjadi lebih mahal.
 - Contoh, Brp gaji yang diterima oleh Joe? ($\text{gaji} = \text{W} * \text{H}$)
 - Untuk nilai-nilai relasi hasil dekomposisi, mungkin kita tidak dapat merekonstruksi nilai-nilai relasi asal yang bersesuaian (*lossless joins*) !
 - Kebetulan tidak terjadi pada contoh relasi SNLRWH
 - Pengecekan beberapa dependensi bisa jadi membutuhkan penggabungan (*joining*) nilai-nilai relasi hasil dekomposisi (*dependency preservation*) !
 - Kebetulan tidak terjadi pada contoh relasi SNLRWH
- Tradeoff: Harus mempertimbangkan issue ini, selain issue redundansi.

Dekomposisi yang Bersifat Lossless Join

- Dekomposisi R menjadi X dan Y disebut *lossless-join* dg mengacu pada satu set FDs F jika, untuk setiap instance r yang memenuhi F, berlaku:
 - $\pi_X(r) \bowtie \pi_Y(r) = r$
- Keadaan yang selalu harus benar: $r \subseteq \pi_X(r) \bowtie \pi_Y(r)$
 - Secara umum, arah sebaliknya tidak berlaku! Jika berlaku, maka dekomposisi bersifat *lossless-join*.
- Definisi di atas dapat secara mudah diperluas utk proses dekomposisi menjadi 3 relasi atau lebih
- *Penting untuk diperhatikan bhw semua jenis dekomposisi yang digunakan untuk menangani redundansi harus bersifat lossless!* (Hindari persoalan ke-2)

Lossless Join (Lanjutan)

- Dekomposisi R menjadi X dan Y bersifat lossless-join dg mengacu pada FDs F, jika dan hanya jika closure dari F (F^+) berisikan:
 - $X \cap Y \rightarrow X$, atau
 - $X \cap Y \rightarrow Y$

The diagram illustrates a lossless join operation. On the left is a 3x3 table with columns A, B, and C. The rows are labeled 1, 4, and 7. The values are: Row 1: A=1, B=2, C=3; Row 4: A=4, B=5, C=6; Row 7: A=7, B=2, C=8. A green arrow points from this table to two smaller 2x3 tables on the right. The top table has columns A and B, with rows 1 and 4. The bottom table has columns B and C, with rows 2, 5, and 2.

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

- Secara umum, dekomposisi R menjadi UV dan R - V bersifat lossless-join jika $U \rightarrow V$ berlaku pada R dan $U \cap V = \emptyset$.

The diagram illustrates a lossless join operation. On the left is a 5x3 table with columns A, B, and C. The rows are labeled 1, 4, 7, 1, and 7. The values are: Row 1: A=1, B=2, C=3; Row 4: A=4, B=5, C=6; Row 7: A=7, B=2, C=8; Row 1: A=1, B=2, C=8; Row 7: A=7, B=2, C=3. A green arrow points from this table to a smaller 3x3 table on the right. The columns are A, B, and C. The rows are labeled 1, 4, and 7. The values are: Row 1: A=1, B=2, C=3; Row 4: A=4, B=5, C=6; Row 7: A=7, B=2, C=8.

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

Dekomposisi yang Mempertahankan Dependensi

- Perhatikan CSJDPQV, C adalah key, $JP \rightarrow C$ dan $SD \rightarrow P$.
 - Dekomposisi BCNF : CSJDQV dan SDP
 - Persoalan: Utk mengecek $JP \rightarrow C$ diperlukan operasi join!
- Dekomposisi yg mempertahankan dependensi (Intuitif):
 - Jika R didekomposisi menjadi X, Y dan Z, dan kita memaksa agar FDs tetap berlaku pada X, Y dan Z, maka semua FDs yang diberikan utk berlaku pada R hrs juga tetap berlaku.
(Menghindari persoalan ke-3)
- Projection dari set FDs F: Jika R didekomposisi menjadi X, ... projection dari F pada X (disimbolkan FX) adalah set dari FDs $U \rightarrow V$ dalam F^+ (*closure of F*) sedemikian rupa sehingga U, V ada dalam X.

Dekomposisi yang Mempertahankan Dependensi (Lanjutan)

- Dekomposisi R menjadi X dan Y bersifat mempertahankan dependensi (dependency preserving) jika $(FX \cup FY) + = F +$, yaitu:
 - Jika kita hanya memperhatikan dependensi dalam $F +$ yang dapat dicek dalam X tanpa memperhatikan Y, dan dalam Y tanpa memperhatikan X, maka hal ini mengimplikasikan bahwa semua dependensi ada dalam $F +$.
- Penting utk memperhatikan $F +$, BUKAN F , dalam definisi ini:
 - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, didekomposisi menjadi AB dan BC.
 - Apakah bersifat *dependency preserving*? apakah $C \rightarrow A$ dipertahankan??
- *Dependency preserving* tidak mengimplikasikan *lossless join*:
 - ABC, $A \rightarrow B$, didekomposisi menjadi AB dan BC.
- Demikian juga sebaliknya, sifat *lossless-join* tidak mengimplikasikan *dependency preserving*

Dekomposisi menjadi BCNF

- Perhatikan relasi R dengan FDs F. Jika $X \rightarrow Y$ melanggar BCNF, lakukan dekomposisi R menjadi $R - Y$ and XY .
 - Penggunaan secara berulang dari ide ini akan menghasilkan sekumpulan relasi yang berada dalam BCNF dan lossless join decomposition, dan dijamin utk mengalami terminasi.
 - Contoh, CSJDPQV, key C, $JP \rightarrow C$, $SD \rightarrow P$, $J \rightarrow S$
 - Utk menangani $SD \rightarrow P$, dekomposisi menjadi SDP, CSJDQV.
 - Utk menangani $J \rightarrow S$, dekomposisi CSJDQV menjadi JS and CJDQV
- Secara umum, beberapa dependensi yang diberikan dapat menimbulkan pelanggaran BCNF. Ingat, urutan “penanganan” dekomposisi seperti di atas dapat memberikan relasi hasil dekomposisi yang berbeda !

- Secara umum, dimungkinkan dekomposisi menjadi BCNF tidak mempertahankan dependensi.
 - Contoh, CSZ, CS → Z, Z → C
 - Tdk dapat didekomposisi utk mempertahankan FD pertama (tidak dpt dilakukan dekomposisi BCNF).
- Dengan cara yang sama, dekomposisi CSJDQV menjadi SDP, JS dan CJDQV tidak mempertahankan dependensi (dengan mengacu ke FDs JP → C, SD → P dan J → S).
 - Namun demikian, dekomposisi di atas bersifat lossless-join.
 - Dalam kasus ini, penambahan JPC pada kumpulan relasi akan memberikan dekomposisi yang dpt mempertahankan dependensi.
 - Penyimpanan tuples JPC hanya untuk tujuan pengecekan FD! (*Persoalan Redundansi!*)

- Algoritma untuk *lossless join decomposition* menjadi BCNF dapat digunakan utk memperoleh *lossless join decomposition* menjadi 3NF (dapat berhenti lebih awal).
- Untuk menjamin *dependency preservation*, suatu ide:
 - Jika $X \rightarrow Y$ tdk dipertahankan, tambahkan relasi XY.
 - Persoalan yang timbul adalah XY dpt melanggar 3NF! Contoh, perhatikan penambahan CJP utk mempertahankan $JP \rightarrow C$. Apa yang terjadi jika juga berlaku $J \rightarrow C$?
- Penyempurnaan: Sebagai pengganti set dari FDs F, gunakan *minimal cover* dari F.

Minimal Cover untuk Set dari FDs

- Minimal cover G utk sebuah set dari FDs F:
 - Closure dari F = closure dari G.
 - Bagian sisi kanan dari setiap FD dalam G berupa sebuah attribut tunggal.
 - Jika G diubah dengan menghapus sebuah FD atau dengan menghapus beberapa attributes dari sebuah FD dalam G, maka closure akan berubah.
- Secara intuitif, setiap FD dalam G diperlukan, dan harus *seminimal mungkin* untuk memperoleh closure yang sama seperti F.
- Contoh, $A \rightarrow B$, $ABCD \rightarrow E$, $EF \rightarrow GH$, $ACDF \rightarrow EG$ mempunyai *minimal cover* berikut:
 - $A \rightarrow B$, $ACD \rightarrow E$, $EF \rightarrow G$ dan $EF \rightarrow H$
- *Minimal Cover* dapat menghasilkan dekomposisi yang bersifat *Lossless-Join* dan *Dependency Preserving*. Decomp !!

Rangkuman

- Jika sebuah relasi berada dalam BCNF, maka relasi tersebut bebas dari redundansi yang dapat dideteksi dengan menggunakan FDs.
 - Dengan demikian, upaya untuk menjamin bahwa semua relasi berada dalam BCNF merupakan upaya heuristik yang baik.
- Jika sebuah relasi tidak berada dalam BCNF, coba lakukan dekomposisi menjadi sekumpulan relasi-relasi BCNF.
 - Harus mempertimbangkan apakah semua FDs dipertahankan. Jika dekomposisi menjadi BCNF yang bersifat lossless-join dan dependency preserving tidak dimungkinkan (atau tidak cocok, untuk beberapa queries yang tipikal), pertimbangkan dekomposisi menjadi 3NF.
 - Dekomposisi sebaiknya dilakukan dan/atau diperiksa kembali dengan mempertimbangkan *performance requirements* yang diinginkan.

PERTEMUAN 10

SQL : Data Manipulation (*Chap. 6 – Conoly*)

TUJUAN DAN PENTINGNYA SQL

SQL adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional.

- SQL adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database
- Idealnya, suatu bahasa database harus memungkinkan user untuk :
 - membuat struktur database dan hubungan (relasi)
 - melakukan tugas dasar mengelola data, seperti penyisipan (insert), perubahan (modify), dan penghapusan (delete) data dari hubungan
 - melakukan pertanyaan sederhana dan rumit

QUERY DALAM SQL

Query adalah bahasa perintah dalam SQL

SQL memiliki 2 komponen utama :

1. *Data Definition Language (DDL)* untuk mendefinisikan struktur database dan mengendalikan akses ke data

- CREATE : membuat tabel atau database
- DROP : menghapus tabel atau database
- ALTER : mengubah struktur tabel, seperti menambah Field (Add), mengganti nama Field (change) atau rename

QUERY DALAM SQL

2. *Data Manipulation Language (DML)* untuk mengambil dan memperbarui data

- INSERT : menginput/memasukkan data pada tabel
- UPDATE : memperbaharui data
- DELETE : menghapus data pada Tabel

SEJARAH SQL

- Tahun 1970
EF Codd → publikasi paper sejarah model relasional
- Tahun 1974
D. Chamberlin → *Structured English Query Language (SEQUEL)*
- Tahun 1976
D. Chamberlin → edisi revisi *SEQUEL/2*, kemudian berubah menjadi SQL untuk alasan hukum
banyak orang masih mengucapkan SQL sebagai "See-Quel", meskipun lafal resminya adalah "S-Q-L".

SEJARAH SQL

- 1976
IBM menghasilkan prototipe DBMS → Sistem *R* akar dari SQL ada di bahasa *SQUARE (Specifying Queries As Relational Expressions)*
- 1981-1983
RDBMS komersial pertama → SQL/DS, untuk DOS/VSE dan VM/CMS, kemudian sebagai DB2 untuk MVS
- 1984
ANSI & ISO → *Relational Database Language (RDL)*
- 1992
ISO → SQL2 atau SQL-92 (ISO, 1992)

Sejarah SQL

- 1999
ISO → SQL:1999 (ISO, 1999a)
- 2003
ISO → SQL: 2003
- 2008
ISO → SQL: 2008



- Sebuah pernyataan SQL terdiri dari :
 - **Reserved words** adalah bagian tetap dari bahasa SQL dan memiliki makna tetap.
 - **User-defined words** dibuat oleh pengguna dengan aturan sintaks tertentu) dan mewakili nama berbagai objek database seperti tabel, kolom, tampilan, indeks, dan sebagainya.

Kebanyakan komponen pernyataan SQL bersifat **case-insensitive**, yang berarti peka terhadap penggunaan huruf besar dan huruf kecil.

Contoh : jika kita menyimpan nama seseorang sebagai "SMITH" dan kemudian mencari dengan menggunakan string "Smith," baris tidak akan ditemukan.



MENULIS PERINTAH SQL

- notasi *Backus Nur Form (BNF)* untuk mendefinisikan pernyataan SQL :
 - huruf besar digunakan untuk mewakili *reserved words* dan harus dieja persis seperti yang ditampilkan;
 - huruf kecil digunakan untuk mewakili user-defined words;
 - sebuah bar vertikal (|) menunjukkan **pilihan** di antara alternatif, misalnya, a b | C;
 - kurung kurawal menunjukkan **elemen yang diperlukan**, misalnya, {a};
 - tanda kurung persegi untuk menunjukkan **elemen opsional**, misalnya, [a];
 - ellipsis (...) Digunakan untuk menunjukkan **pengulangan** opsional item nol atau lebih.

MANIPULASI DATA

- pernyataan *Data Manipulation Language (DML)* SQL:
 - SELECT - untuk query data dalam database;
 - INSERT - untuk memasukkan data ke dalam tabel;
 - UPDATE - untuk memperbarui data dalam tabel;
 - DELETE - menghapus data dari tabel.
- Semua nilai data *nonnumeric* harus diapit tanda kutip tunggal, semua nilai data *numerik* tidak harus diapit tanda kutip tunggal.

Contoh :

```
INSERT INTO PropertyForRent(propertyNo, street, city, postcode,
    type, rooms, rent, ownerNo, staffNo, branchNo)
VALUES ('PA14', '16 Holhead', 'Aberdeen', 'AB7 5SU', 'House', 6,
    650.00, 'CO46', 'SA9', 'B007');
```

QUERY SEDERHANA

- Tujuan dari pernyataan SELECT adalah untuk mengambil dan menampilkan data dari satu atau lebih tabel database.
- SELECT adalah perintah SQL yang paling sering digunakan dan memiliki bentuk umum sebagai berikut :

```
SELECT      [DISTINCT | ALL] { * | [columnExpression]
            [AS newName]] [, . . . ])
FROM        TableName [alias] [, . . . ]
[WHERE      condition]
[GROUP BY   columnList] [HAVING condition]
[ORDER BY   columnList]
```

QUERY SEDERHANA

- Urutan pengolahan dalam sebuah pernyataan SELECT adalah :
FROM spesifik tabel atau tabel yang akan digunakan
WHERE filter baris subjek pada beberapa kondisi
GROUP BY bentuk kelompok baris dengan nilai kolom yang sama
HAVING filter kelompok subjek pada beberapa kondisi
SELECT menentukan kolom mana yang muncul dalam output
ORDER BY menentukan urutan output

CONTOH QUERY SEDERHANA

cara cepat untuk mengungkapkan "semua kolom" di SQL, dengan menggunakan tanda bintang (*)

- **CONTOH : Mengambil semua kolom, semua baris**

Daftar lengkap rincian semua staf.

```
SELECT staffNo, fName, lName, position, sex, DOB, salary, branchNo
FROM Staff;
```

- Cara cepat :

```
SELECT *
FROM Staff;
```

CONTOH QUERY SEDERHANA

- **Tabel Hasil**

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	01-Okt-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

CONTOH QUERY SEDERHANA

- **CONTOH 6.2 Mengambil kolom tertentu, semua baris**

Menghasilkan daftar gaji untuk semua staf hanya menampilkan jumlah staf, nama pertama dan terakhir, dan rincian gaji.

SELECT staffNo, fName, lName, salary
FROM Staff;

Tabel Hasil

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

MENGGUNAKAN FUNGSI AGREGAT SQL

Standar ISO mendefinisikan lima fungsi agregat :

- COUNT — menghasilkan jumlah nilai dalam sebuah kolom tertentu
- SUM — menghasilkan jumlah nilai dalam kolom tertentu
- AVG — menghasilkan rata-rata nilai dalam sebuah kolom tertentu
- MIN — menghasilkan nilai terkecil pada kolom yang ditentukan
- MAX — menghasilkan nilai terbesar dalam satu kolom tertentu

MENGGUNAKAN COUNT(*)

- **CONTOH Menggunakan COUNT(*)**

Berapa banyak biaya properti lebih dari £350 per bulan untuk menyewa?

```
SELECT COUNT(*) AS myCount
FROM PropertyForRent
WHERE rent >350;
```

- **Contoh Menggunakan COUNT(DISTINCT)**

Berapa banyak properti yang berbeda ditampilkan bulan Mei 2008?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount
```

```
FROM Viewing
```

```
WHERE viewDate BETWEEN '1-May-08' AND '31-May-08';
```

MENGGUNAKAN COUNT DAN SUM

- **Contoh Menggunakan COUNT dan SUM**

Mencari jumlah Manajer dan jumlah gaji mereka.

```
SELECT COUNT(staffNo) AS myCount, SUM(salary) AS mySum
FROM Staff
WHERE position = 'Manager';
```

Tabel Hasil

myCount	mySum
2	54000.00

MENGGUNAKAN MIN, MAX, AVG

- CONTOH Menggunakan MIN, MAX, AVG

Mencari minimum, maksimum, dan rata-rata gaji staf.

```
SELECT      MIN(salary) AS myMin, MAX(salary) AS myMax,  
AVG(salary) AS myAvg  
FROM Staff;
```

- TABEL Hasil

myMin	myMax	myAvg
9000.00	30000.00	17000.00

MEMPERBAHARUI DATABASE

Tiga pernyataan SQL yang tersedia untuk memodifikasi isi dari tabel dalam database :

- INSERT : menambah baris baru dari data ke tabel
- UPDATE : memodifikasi data yang ada dalam tabel
- DELETE : menghapus baris data dari tabel

MENAMBAHKAN DATA KE DALAM DATABASE (INSERT)

- Ada dua bentuk pernyataan INSERT.
- Yang pertama memungkinkan satu baris untuk dimasukkan ke tabel bernama dan memiliki format berikut :
INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

CONTOH INSERT... VALUES

Menyisipkan baris baru ke dalam tabel Staff untuk memasok data untuk semua kolom.

INSERT INTO Staff

VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M', **DATE** '1957-05-25', 8300, 'B003');

MENAMBAHKAN DATA KE DALAM DATABASE (INSERT)

- Bentuk kedua dari pernyataan INSERT memungkinkan beberapa baris untuk disalin dari satu atau lebih tabel yang lain, dan memiliki format berikut :
- **INSERT INTO** TableName [(columnList)]
SELECT...

CONTOH INSERT... SELECT

```
INSERT INTO StaffPropCount
(SELECT s.staffNo, fName, lName, COUNT(*)
     FROM Staff s, PropertyForRent p
    WHERE s.staffNo = p.staffNo
  GROUP BY s.staffNo, fName, lName)
UNION
(SELECT staffNo, fName, lName, 0
     FROM Staff s
    WHERE NOT EXISTS (SELECT *
                          FROM PropertyForRent p
                         WHERE p.staffNo = s.staffNo));
```

MODIFIKASI DATA PADA DATABASE (UPDATE)

Pernyataan UPDATE memungkinkan isi baris yang ada di tabel bernama diubah. Format, perintah adalah :

UPDATE TableName

SET columnName1 = dataValue1 [,columnName2 = dataValue2 . . .]
[WHERE searchCondition]

Jika klausa WHERE diterapkan, hanya baris yang memenuhi *searchCondition* yang diperbarui. Para *dataValue* baru harus sesuai dengan tipe data untuk kolom yang sesuai.

CONTOH UPDATE semua baris

Berikan semua staf kenaikan gaji 3%.

UPDATE Staff

SET salary = salary*1.03;

④ CONTOH UPDATE baris spesifik

Berikan Manager minyak kenaikan gaji 5%.

UPDATE Staff

SET salary = salary*1.05

WHERE position = 'Manager';

⑤ CONTOH UPDATE banyak kolom

Promosikan David Ford (*staffNo* = 'SGI4') untuk Manager dan mengubah gajinya menjadi £18.000.

UPDATE Staff

SET position = 'Manager', salary = 18000

VHERE staffNo = 'SGI4';

Menghapus Data dari Database (DELETE)

- Pernyataan **DELETE** memungkinkan baris yang akan dihapus dari tabel bernama. Format perintahnya adalah :
DELETE FROM TableName
[WHERE searchCondition]
- **CONTOH DELETE** baris yang spesifik
Hapus semua tampilan yang berhubungan dengan properti PG4.
DELETE FROM Viewing
WHERE propertyNo = 'PG4';
- **CONTOH DELETE** semua baris
Hapus semua baris dari tabel Viewing.
DELETE FROM Viewing;



PERTEMUAN 11

SQL (*lanjutan*): Queries, Constraints & Triggers

(Chap. 5 – Ramakrishnan)

Overview

Structure Query Language (SQL) adalah bahasa database relasional komersial yang paling banyak digunakan. SQL pada awalnya dikembangkan oleh IBM dalam SEQUEL-XRM dan Proyek System-R (1974-1977). Kemudian SQL berkembang mengikuti standar ANSI/ISO untuk SQL, yang disebut SQL-92.

Beberapa Aspek Bahasa SQL

- Data Definition Language (DDL) : subset SQL yang mendukung pembuatan, penghapusan, dan modifikasi struktur tabel beserta tampilannya.
- Data Manipulation Language (DML) : subset SQL dapat digunakan untuk menspesifikasikan queries, menyisipkan, menghapus, dan memodifikasi baris-baris tabel.
- Embedded dan Dinamic SQL : Fitur-fitur embedded SQL yang memungkinkan SQL untuk memanggil host language seperti C atau COBOL.
- Triggers : Standar SQL/1999 memberikan dukungan untuk triggers, yang bertindak secara otomatis dan memanipulasi database ketika kondisi terpenuhi.

- Security : SQL menyediakan mekanisme untuk mengendalikan akses pengguna ke objek database seperti tables dan views.
- Transaction Management : perintah SQL yang memungkinkan seseorang pengguna melakukan secara eksplisit untuk mengendalikan aspek, bagaimana sebuah transaksi harus dijalankan.
- Client-Server Execution & Remote Database Access : perintah-perintah SQL ini dapat digunakan untuk mengendalikan bagaimana suatu program aplikasi dapat dihubungkan ke sebuah SQL database server, atau mengakses data dari sebuah database melalui jaringan.

Contoh Instance

B1

bid	bname	color
101	Interlake	Blue
103	Clipper	Green

B2

bid	bname	color
102	Interlake	Red
104	Marine	Red

S1

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Contoh Instance

R1

Sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

S2

sid	sname	rating	age
28	Yuppy	9	35.0
31	Lubber	8	55.5
44	Guppy	5	35.0
58	Rusty	10	35.0

R2

Sid	bid	day
22	101	10/10/96
58	103	10/12/96

Bentuk Dasar SQL Query

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
```

- *FROM-list* : Sebuah nama tabel dapat diikuti oleh berbagai variabel yang sangat berguna ketika nama tabel yang sama muncul lebih dari sekali dalam daftar.
- *Select-list* : list dari (ekspresi yang melibatkan) nama kolom atau field dari tabel.
- *Qualification* : klausula WHERE kombinasi boolean (AND, OR, dan NOT) di dalam bentuk ekspresi op ekspresi, dimana op adalah salah satu operator perbandingan {<, <=, =, >, >=, >}.
- *DISTINCT* : keyword yang opsional. Hal ini menunjukkan bahwa tabel yang dihitung sebagai hasil dan tidak harus mengandung *duplicate*, yaitu dua baris data yang sama. Defaultnya adalah *duplicate* yang tidak dihilangkan.

Contoh Basic SQL Query

- Tampilkan nama dan umur dari tabel Sailors :

```
SELECT DISTINCT S.sname,S.age
FROM Sailors S
```

The screenshot shows the phpMyAdmin interface. In the top left, there's a logo for STMIK NUSA MANDIRI. The main area has two panes: one for writing SQL queries and another for running them.

SQL query:

```
SELECT S.sname, S.age
FROM S
LIMIT 0 , 30
```

Run SQL query/queries on database sql:

```
SELECT S.sname, S.age
FROM S
```

Fields:

sid	bid	day
-----	-----	-----

Results:

	sname	age
[checkbox]	Dustin	45
[checkbox]	Brutus	33
[checkbox]	Lubber	55.5
[checkbox]	Andy	25.5
[checkbox]	Rusty	35
[checkbox]	Horatio	35
[checkbox]	Zorba	16
[checkbox]	Horatio	35
[checkbox]	Art	25.5
[checkbox]	Bob	63.5

Below the results, there are buttons for "Check All / Uncheck All With selected:" and other navigation controls like "Show:", "mode and repeat headers after 100 cells", and "Print view".

Strategi Evaluasi Konseptual

- Hitung hasil cross product dari form list.
- Hapus tuples hasil jika tuples tersebut tidak memenuhi qualifications.
- Hapus attributes yang tidak ada dalam select list.
- Jika digunakan DISTINCT, lakukan eliminasi baris-baris yang terduplicasi.

Contoh Strategi Konseptual :

- Tentukan nama nama pelaut yang telah memesan sejumlah 103 perahu.

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

The screenshot shows the phpMyAdmin interface for a database named 'sql'. The left sidebar lists databases 'b', 'r', and 'sql (3)'. The main area displays a query result for rows 0-2 of a total of 3. The SQL query is:

```
SELECT S.sname  
FROM S, R  
WHERE S.sid = R.sid  
AND R.bid = 103  
LIMIT 0, 30
```

The results table shows three rows:

sname
Dustin
Lubber
Horatio

Below the table are buttons for 'Insert new row', 'Print view', 'Print view (with full texts)', and 'Export'.

At the bottom, the footer includes the text 'Program Studi Sistem Informasi', 'STMIK NUSA MANDIRI', and 'COPYRIGHT (C) Sept 2012'.

The screenshot shows the phpMyAdmin interface for a database named 'sql'. The left sidebar lists databases 'b', 'r', and 'sql (3)'. The main area displays a query result for rows 0-0 of a total of 1. The SQL query is:

```
SELECT S.age  
FROM Sailors S  
WHERE S.sname LIKE 'B_%B'  
LIMIT 0, 30
```

The results table shows one row:

age
63.5

Below the table are buttons for 'Insert new row', 'Print view', 'Print view (with full texts)', and 'Export'.

At the bottom, the footer includes the text 'Program Studi Sistem Informasi', 'STMIK NUSA MANDIRI', and 'COPYRIGHT (C) Sept 2012'.

Union, Intersect & Except

- UNION : dapat digunakan untuk menghitung union dari dua union-compatible yang merupakan kumpulan record dari hasil dua queries.
- INTERSECT : dapat digunakan untuk menghitung intersect dari dua intersect yang merupakan kumpulan dari record.
- Menampilkan name dari Sailors yang telah melakukan reservasi sebuah red boat atau green boat.

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
AND (B.color = 'red' OR B.color = 'green')
```

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
B.color = 'red'
UNION
SELECT S2.sname
FROM Sailors S2, Boats B2, Reserves R2
WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND
B2.color = 'green'
```

Union, Intersect & Except

- EXCEPT : dapat digunakan untuk menghitung *set difference* dari dua union-compatible yang merupakan kumpulan record dari hasil dua queries.

- Cari *sids* dari semua pelaut yang telah memesan Boat red tetapi tidak memesan Boat green.

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
B.color = 'red'
INTERSECT
SELECT S2.sname
FROM Sailors S2, Boats B2, Reserves R2
WHERE S2.sid = R2.sid AND R2.bid = B2.bid
AND B2.color = 'green'
```

```
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
B.color = 'red'
EXCEPT
SELECT S.sid
FROM Sailors S2, Reserves R2, Boats B2
WHERE S2.sid = R2.sid AND R2.bid = B2.bid
AND B2.color = 'green'
```

Nested Queries

- Tentukan nama-nama pelaut yang telah memesan boat bernomor 103.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECT R.sid
                  FROM Reserves R
                  WHERE R.bid = 103 )
```

- Operator IN memungkinkan kita untuk menguji apakah nilai dalam himpunan elemen; sebuah query SQL yang digunakan untuk menghasilkan data pada query yang akan diuji.
- Untuk menampilkan name pada tabel Sailors yang tidak melakukan reservasi boat bernomor 103 gunakan NOT IN.

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** sql (3)
- Navigation:** localhost > sql
- Toolbar:** Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Empty, Drop
- SQL Query:**

```
SELECT S.sname
FROM Sailors S
WHERE S.sid
IN (
    SELECT R.sid
    FROM Reserves R
    WHERE R.bid = 103
)
LIMIT 0 , 30
```
- Result Panel:** Shows the results of the executed query:

sname
Dustin
Lubber
Horatio
- Run SQL Query:**

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECT R.sid FROM Reserves R
WHERE R.bid = 103 )
```

 Show this query here again

Korelasi Nested Queries

- Tentukan nama-nama sailors yang telah memesan boat bernomor 103

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS ( SELECT * FROM Reserves R
                WHERE R.bid = 103 AND R.sid = S.sid )
```

- EXIST** : bentuk operator perbandingan yang jika bernilai TRUE akan dijadikan subquery sebagai parameter yang tidak menghilangkan set kosong.
- Untuk menampilkan name pada tabel Sailors yang tidak memesan boat bernomor 103 dengan menggunakan NOT EXIST.

The screenshot shows two windows of the phpMyAdmin interface. The left window displays the results of a query:

```
Showing rows 0 - 2 (3 total, Query took 0.0016 sec)
+-----+
| sname |
+-----+
| Dustin |
| Lubber |
| Horatio |
+-----+
```

The right window shows the SQL query being run in a separate window:

```
http://localhost/phpmyadmin/querywindow.php?lang=en-utf8&server=1&collation_connection=utf8_general_ci
SQL Import files SQL history
Run SQL query/queries on database sql: 
SELECT S.sname
FROM S WHERE EXISTS (SELECT * FROM R WHERE R.bid = "103"
AND R.sid = S.sid )
LIMIT 0 , 30
Fields
sid
bid
day
<<
```

Below the query window, there are checkboxes for "Do not overwrite this query from outside the window" and "Show this query here again".

Set-Comparison Operators

- Sebelumnya telah dibahas penggunaan EXIST, IN, dan UNIQUE. SQL juga mendukung op ANY dan op ALL, dimana op adalah salah satu operator perbandingan aritmatika {<, <=, =, >, >=}. (SOME juga tersedia, tapi itu hanya sinonim untuk ANY)
- Contoh : Cari sailors yang mempunyai rating yang lebih besar dari sailors Horatio

```
SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                      FROM Sailors S2
                      WHERE S2.sname = 'Horatio')
```

Division dalam SQL

- Tentukan nama sailors yang telah melakukan reservasi semua boats.
- Dengan menggunakan EXCEPT :

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (( SELECT B.bid FROM Boats B )
EXCEPT
(SELECT R.bid FROM Reserves R
 WHERE R.sid = S.sid ))
```

- Cara yang lebih sulit tanpa menggunakan EXCEPT :

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS ( SELECT B.bid FROM Boats B
WHERE NOT EXISTS ( SELECT R.bid
FROM Reserves R
WHERE R.bid = B.bid AND
R.sid = S.sid ))
```

localhost > sql > R

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Showing rows 0 - 0 (total, Query took 0.0639 sec)

SQL query:

```

SELECT S.sname
FROM S
WHERE NOT
EXISTS (
    SELECT B.bid
    FROM B
    WHERE NOT
    EXISTS (
        SELECT R.bid
        FROM R
        WHERE R.bid = B.bid
        AND R.sid = S.sid
    )
)
LIMIT 0 , 30

```

Run SQL query/queries on database **sql**

Fields

sid
bid
day

Done

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

sname

Dustin

Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

STMIK NUSA MANDIRI

COPYRIGHT (C) Sept 2012

Operator Agregasi

- COUNT ([DISTINCT] A) : Jumlah dari semua nilai-nilai dalam kolom A.
- SUM ([DISTINCT] A) : Jumlah dari semua (unik) nilai-nilai dalam kolom A.
- AVG ([DISTINCT] A) : Rata-rata dari semua (unik) nilai-nilai dalam kolom A.
- MAX (A) : Nilai maksimum di kolom A.
- MIN (A) : Nilai minimum dalam kolom A.

```
SELECT COUNT (*)
FROM Sailors S
```

```
SELECT COUNT (DISTINCT S.sname)
FROM Sailors S
```

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

```
SELECT S.sname, S.age
FROM Sailors S
WHERE S.age=(SELECT MAX (S2.age)
FROM Sailors S2)
```

Program Studi
Sistem Informasi

STMIK NUSA MANDIRI

COPYRIGHT (C) Sept 2012

Hasil dari : SELECT COUNT (*) FROM Sailors S

localhost > sql > S

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Showing rows 0 - 9 (10 total, Query took 0.0004 sec)

SQL query:

```
SELECT COUNT(*)
FROM S
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show: 30 row(s) starting from record #0
in horizontal mode and repeat headers after 100 cells
Sort by key: None Go

COUNT(*)
10

Show: 30 row(s) starting from record #0
in horizontal mode and repeat headers after 100 cells

[Print view](#) [Print view \(with full texts\)](#) [Export](#)

phpMyAdmin - Mozilla Firefox
<http://localhost/phpmyadmin/querywindow.php?lang=en-utf8&>
 Run SQL query/queries on database sql: ②
 SELECT COUNT(*)
 FROM S
 Fields sid sname
 Done

Hasil dari : SELECT AVG (S.age) FROM Sailors S WHERE S.rating=10

localhost > sql > S

Browse Structure SQL Search Insert Export Import Operations Empty Drop

Showing rows 0 - 0 (1 total, Query took 0.0006 sec)

SQL query:

```
SELECT AVG(S.age)
FROM S
WHERE S.rating = 10
```

Show: 30 row(s) starting from record #0
in horizontal mode and repeat headers after 100 cells
AVG (S.age)
25.5

Show: 30 row(s) starting from record #0
in horizontal mode and repeat headers after 100 cells

[Print view](#) [Print view \(with full texts\)](#) [Export](#)

phpMyAdmin - Mozilla Firefox
<http://localhost/phpmyadmin/querywindow.php?lang=en-utf8&>
 Run SQL query/queries on database sql: ②
 SELECT AVG (S.age)
 FROM S
 WHERE S.rating=10
 Fields sid sname rating age
 Done

Hasil dari : `SELECT S.sname, S.age FROM Sailors S
WHERE S.age=(SELECT MAX (S2.age) FROM Sailors S2)`

sname	age
Lubber	55.5

GROUP BY & HAVING CLAUSE

- ❑ Sejauh ini telah dibahas penggunaan operasi agregasi untuk semua tuples (yang memenuhi kualifikasi). Seringkali kita ingin menerapkan operasi agregasi untuk sejumlah kelompok (groups) dari baris (tuples) dalam suatu relasi.
- ❑ Contoh : Cari usia sailors termuda untuk setiap tingkat rating yang ada.

```
SELECT MIN(S.age)
      FROM Sailors S
     WHERE S.rating=i
```

Jika misalnya, kita ketahui bahwa nilai rating berada dalam range 1 s.d 10, maka i pada pernyataan diatas = 1,2,3,...,10.

Queries dengan GROUP BY & HAVING

- Bentuk umum SQL Query dengan GROUP BY & HAVING

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

- Contoh : Carilah usia sailors termuda yang memenuhi syarat (misalnya, setidaknya ≥ 18 tahun) untuk setiap tingkat rating dengan setidaknya terdiri dari dua sailors untuk setiap tingkat ratingnya.

```
SELECT S.rating, MIN (S.age) AS minage
      FROM Sailors S
     WHERE S.age >=18
      GROUP BY S.rating
        HAVING COUNT (*) >1
```

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** sql (3)
- SQL query:**

```
SELECT S.rating, MIN(S.age) AS minage
  FROM S
 WHERE S.age >=18
  GROUP BY S.rating
    HAVING COUNT(*) >1
  LIMIT 0, 30
```
- Result:** The results table displays three rows of data:

	rating	minage
<input type="checkbox"/>	3	25.5
<input type="checkbox"/>	7	35
<input type="checkbox"/>	8	25.5
- PHP Window:** An inset window titled "phpMyAdmin - Mozilla Firefox" shows the same SQL query and results, indicating the query was run in a separate window.

NULL VALUES

- Nilai-nilai fields dalam sebuah tuple kadang-kadang tidak diketahui (unknown). Misalnya : sebuah nilai rating tidak diberikan atau tidak dapat digunakan (inapplicable). Maka SQL menyediakan nilai kolom khusus yang disebut null untuk digunakan dalam situasi tersebut.

NULL VALUES dan Operator Perbandingan serta Logical Connectives AND, OR, NOT

- SQL menyediakan operator perbandingan khusus IS NULL untuk menguji apakah kolom nilai nol yang akan mengevaluasi dengan benar pada AND yang mewakili baris. Disini juga terdapat IS NOT NULL, yang akan mengevaluasi nilai false pada baris untuk AND.

Dampak NULL VALUES dalam Membangun SQL

- Untuk kualifikasi dalam klausa WHERE clause, keberadaan null values dapat menghilangkan baris (dalam garis-produk dari tabel disebutkan dalam klausa FROM) yang kualifikasi tidak mengevaluasi nilai TRUE.
 - Menghilangkan baris yang mengevaluasi unknown mempunyai dampak yang halus namun signifikan pada queries, terutama nested queries yang melibatkan EXISTS atau UNIQUE.
- Persoalan lain adalah definisi SQL yang menyatakan bahwa dua baris duplikat jika kolom yang sesuai adalah sama baik, atau keduanya bersifat null. Dalam kenyataannya jika kita membandingkan dua nilai null menggunakan =, hasilnya adalah unknown! Dalam konteks duplikat, perbandingan ini secara implisit diperlakukan sebagai nilai true, yang merupakan anomali.
- Operator aritmatika +, -, *, dan / semua menghasilkan nilai null jika salah satu dari argumennya bernilai null.

Dampak NULL VALUES dalam Membangun SQL (Lanjutan)

- Null Values dapat menimbulkan hal yang tidak diharapkan untuk operator-operator agregasi :
 - COUNT (*) menangani nilai null seperti halnya nilai-nilai lainnya (ikut diperhitungkan).
 - Operasi-operasi agregasi lainnya (COUNT, SUM, AVG, MIN, MAX, dan variasi penggunaan DISTINCT) hanya mengabaikan null values.

OUTER JOINS

- Beberapa varian menarik dari operasi join yang mengandalkan null values disebut **outer joins**.
- Terdapat tiga variasi outer join :
 1. Left Outer Join
 2. Right Outer Join
 3. Full Outer Join
- Sebagai contoh, query berikut adalah daftar sid, pasangan sesuai dengan pelaut dan mereka yang telah memesan perahu:

```
SELECT Sailors.sid, Reserves.bid  
FROM Sailors NATURAL LEFT OUTER JOIN Reserve R
```

The screenshot shows the phpMyAdmin interface. On the left, there's a sidebar with icons for Home, Import, Export, Query, and Help. Below that is a dropdown menu for 'Database' set to 'sql (3)'. Under the database name, there are three tables: 'b', 'r', and 's'. The main area has two tables. The top table, titled 'sid bid', contains rows of data. The bottom table, also titled 'sid bid', contains a SQL query and its results. The SQL query is:

```
SELECT S.sid, R.bid  
FROM S NATURAL LEFT OUTER JOIN R
```

The results show fields sid, sname, rating, and age.

Tidak Membolehkan Null Values

- Melarang nilai bersifat null dengan menetapkan NOT NULL sebagai bagian dari definisi sebuah field, misalnya :
`sname CHAR (20) NOT NULL`
- Selain itu, field dalam PRIMARY KEY tidak diperbolehkan bernilai null. Dengan demikian, ada kendala penggunaan NOT NULL secara implisit untuk setiap field yang tercantum dalam PRIMARY KEY.

PERTEMUAN 12

Keamanan dan Administrasi Database

(Chap. 20 – Conolly)

Keamanan Database

Keamanan Database : Mekanisme yang melindungi *database* terhadap ancaman disengaja atau tidak disengaja.

Keamanan database dalam kaitannya dengan situasi berikut:

1. pencurian dan penipuan;
2. hilangnya kerahasiaan;
3. hilangnya privasi;
4. hilangnya integritas;
5. hilangnya ketersediaan.

Ancaman

- **Ancaman** : Setiap situasi atau peristiwa, baik disengaja atau tidak disengaja, yang bisa mempengaruhi sistem dan akibatnya organisasi.

TABEL Contoh ancaman.

TABLE 20.1 Examples of threats.

THREAT	THEFT AND FRAUD	LOSS OF CONFIDENTIALITY	LOSS OF PRIVACY	LOSS OF INTEGRITY	LOSS OF AVAILABILITY
Using another person's means of access	✓	✓		✓	
Unauthorized amendment or copying of data	✓			✓	
Program alteration	✓			✓	✓
Inadequate policies and procedures that allow a mix of confidential and normal output	✓	✓		✓	
Wire tapping	✓	✓		✓	
Illegal entry by hacker	✓	✓		✓	
Blackmail	✓	✓		✓	
Creating "trapdoor" into system	✓	✓		✓	
Theft of data, programs and equipment	✓	✓		✓	✓
Failure of security mechanisms, giving greater access than normal		✓	✓	✓	
Staff shortages or strikes				✓	✓
Inadequate staff training	✓	✓	✓	✓	
Viewing and disclosing unauthorized data	✓	✓		✓	
Electronic interference and radiation				✓	✓
Data corruption owing to power loss or surge				✓	✓
Fire (electrical fault, lightning strike, arson), flood, bomb				✓	✓
Physical damage to equipment				✓	✓
Breaking cables or disconnection of cables				✓	✓
Introduction of viruses				✓	✓

Gambar Ringkasan potensi ancaman sistem komputer.

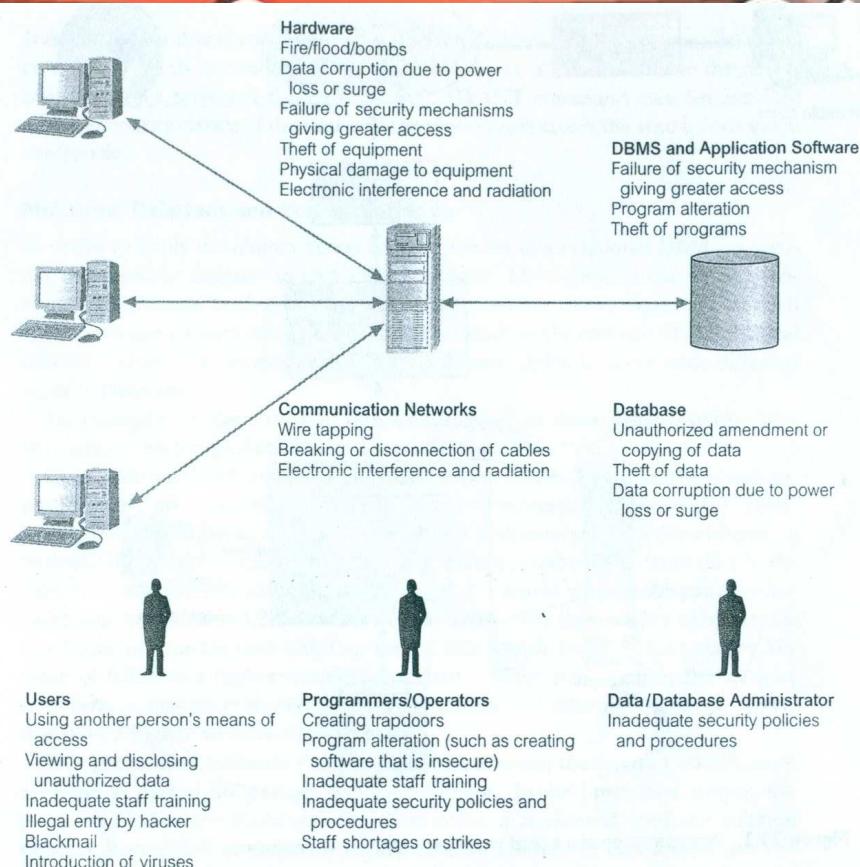


Figure 20.1 Summary of potential threats to computer systems.

Penanggulangan-Komputer Berbasis Kontrol

Keamanan untuk lingkungan *multi-user* (beberapa di antaranya mungkin tidak tersedia di lingkungan PC):

1. Otorisasi dan otentikasi

Otorisasi :

Pemberian hak atau hak istimewa yang memungkinkan subjek untuk memiliki akses yang sah ke sistem atau objek sistem

Otentikasi :

Sebuah mekanisme yang menentukan apakah seorang pengguna bertanggung jawab untuk mengakses komputer dengan menciptakan *account* individu. Dimana setiap *user* diberikan pengenal unik, yang digunakan oleh sistem operasi untuk menentukan siapa mereka.

2. Akses kontrol : DAC, MAC

Akses kontrol untuk sistem *database* didasarkan pada pemberian dan pencabutan hak-hak istimewa. Sebuah **hak istimewa** memungkinkan pengguna untuk membuat atau akses (yaitu membaca, menulis, atau memodifikasi) beberapa objek *database* (seperti relasi, melihat, atau indeks) atau untuk menjalankan utilitas tertentu DBMS.

Discretionary Access Control (DAC)

DBMS yang paling komersial menyediakan pendekatan untuk mengelola hak istimewa yang menggunakan SQL *Discretionary Access Control* disebut (DAC). Standar SQL mendukung DAC melalui GRANT dan REVOKE perintah. Perintah GRANT memberikan hak istimewa kepada pengguna, dan perintah REVOKE menghapus hak istimewa.

Mandatory Access Control (MAC)

Dalam pendekatan ini setiap objek *database* diberikan sebuah keamanan kelas dan setiap pengguna diberikan izin untuk kelas keamanan, dan aturan dikenakan pada membaca dan menulis objek *database* oleh pengguna

3. Views,

adalah hasil dinamik dari satu atau lebih operasi relasional operasi pada relasi untuk menghasilkan relasi lainnya. View adalah relasi virtual yang tidak benar-benar ada dalam database, tetapi dihasilkan atas permintaan pengguna tertentu, pada saat ada nya permintaan.

Mekanisme Tampilannya menyediakan keamanan yang kuat dan fleksibel dengan menyembunyikan bagian-bagian dari database dari pengguna tertentu.

4. Backup dan Journal,

Backup : Proses periodik menyalin database dan file log (dan mungkin program) ke media penyimpanan offline.

Journal : Proses memelihara sebuah file log (atau jurnal) dari semua perubahan yang dibuat oleh database secara efektif .

5. Enkripsi,

Pengkodean data dengan algoritma khusus yang membuat data terbaca oleh program tanpa kunci dekripsi.

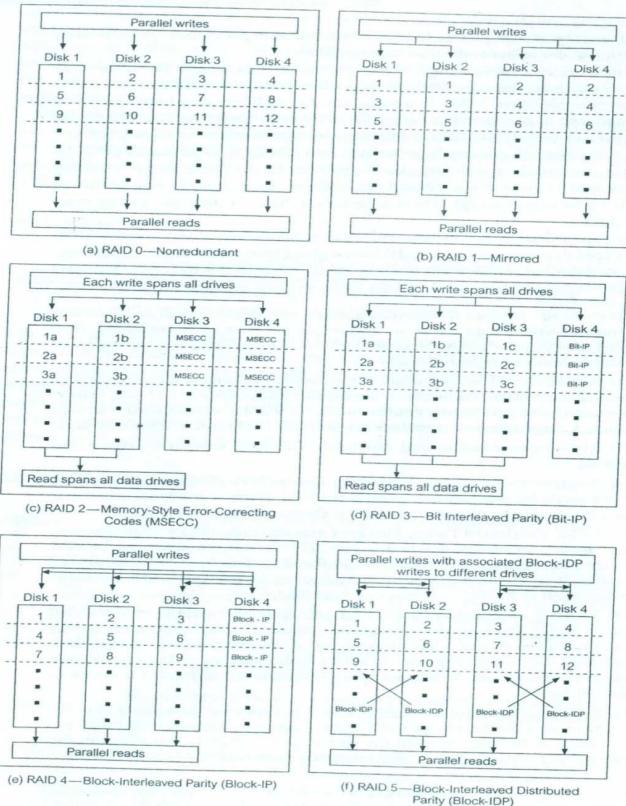
Beberapa DBMS menyediakan fasilitas enkripsi yang dapat mengakses data (setelah decoding itu), meskipun ada degradasi dalam performa karena waktu yang dibutuhkan untuk memecahkan kode tersebut . Enkripsi juga melindungi data yang dikirimkan melalui. jalur komunikasi.

6. RAID Teknologi.

RAID awalnya berdiri untuk *Redundant Array of Independent Disk*.

RAID bekerja pada sebuah array disk besar terdiri dari susunan beberapa disk yang diselenggarakan untuk meningkatkan kehandalan dan kinerja waktu pada tingkatan yang sama.

Figure 20.4
RAID levels.
The numbers represent sequential data blocks and the letters indicate segments of a data block.



RAID tingkat. Angka-angka mewakili blok sekuensial data dan surat-surat menunjukkan segmen blok data

STMIK NUSA MANDIRI

COPYRIGHT (C) Sept 2012

Keamanan di Microsoft Office Access DBMS

Microsoft Office Access menyediakan metode berikut untuk mengamankan database:

1. Memisahkan database;

Cara yang paling aman untuk melindungi data dalam database adalah untuk menyimpan tabel database terpisah dari objek aplikasi database seperti formulir dan laporan. Tindakan ini disebut sebagai "pemisahan" database;

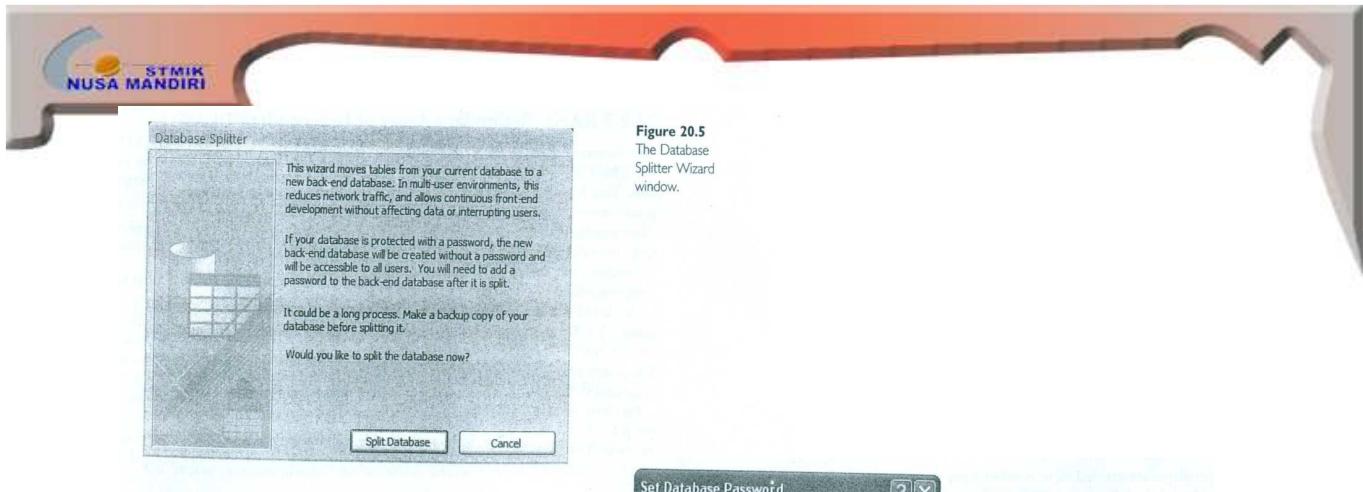


Figure 20.5
The Database Splitter Wizard window.

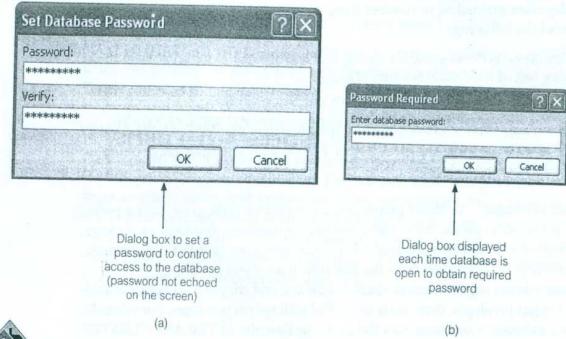


Figure 20.6 Securing the DreamHome database using a password: (a) the Set Database Password dialog box; (b) the Password Required dialog box shown at startup.

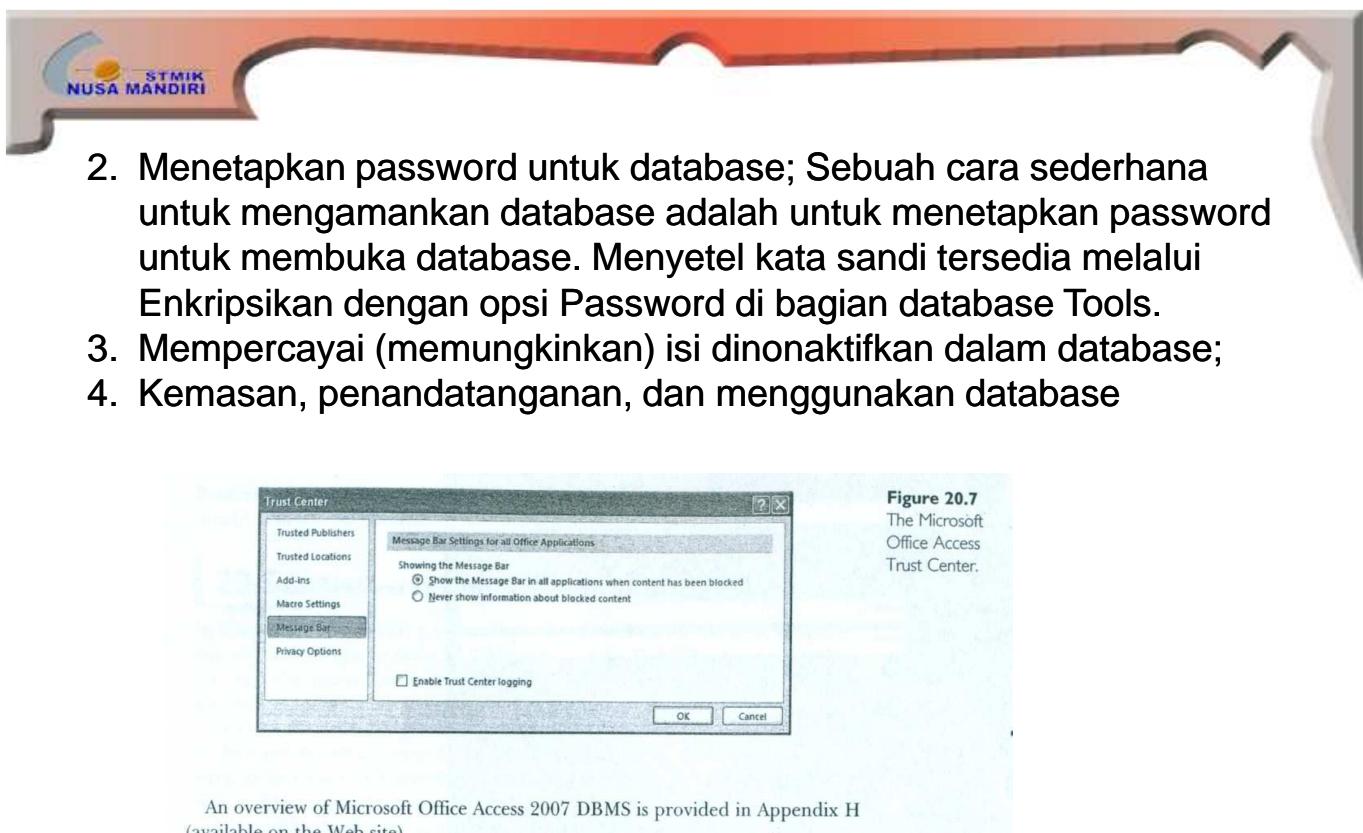


Figure 20.7
The Microsoft Office Access Trust Center.

An overview of Microsoft Office Access 2007 DBMS is provided in Appendix H (available on the Web site).



Keamanan di Oracle DBMS

- **Keistimewaan :**

Beberapa contoh hak Oracle mencakup hak untuk : Terhubung ke database (membuat sesi); Membuat tabel; Pilih baris dari tabel pengguna lain.

Dalam Oracle, ada dua kategori yang berbeda dari hak istimewa : Sistem hak istimewa; Obyek hak istimewa.

- **Sistem hak istimewa**

Hak istimewa sistem yang diberikan kepada, atau dicabut dari, pengguna dan peran (dibahas di bawah) menggunakan salah satu dari berikut:

- Hibah Keistimewaan Sistem/kotak Peran dialog dan Mencabut KeistimewaanSistem/Peran kotak dialog Manajer Keamanan Oracle;
- SQL GRANT dan laporan REVOKE

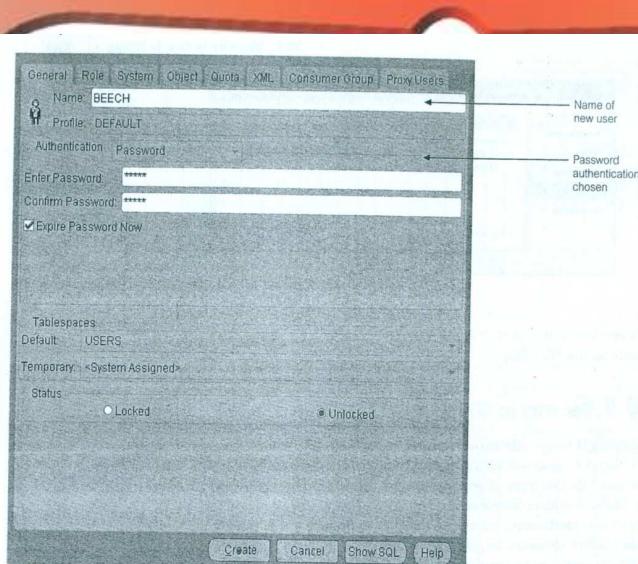
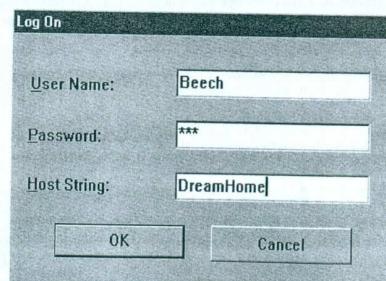


Figure 20.8 Creation of a new user called Beech with password authentication set.

Figure 20.9
Log On dialog box requesting user name, password, and the name of the database the user wishes to connect to.



1. Proxy Server :

Dalam lingkungan Web, proxy server adalah sebuah komputer yang berada di antara Web browser dan server Web.

Proxy server memiliki dua tujuan utama:

- a. **Meningkatkan kinerja**, yaitu proxy server menyimpan hasil dari semua permintaan untuk jumlah waktu tertentu secara signifikan
- b. **Filter permintaan** yaitu Proxy server dapat digunakan untuk menyaring permintaan. Sebagai contoh, sebuah organisasi yang menggunakan server proxy untuk mencegah karyawan mengakses satu set spesifik situs Web.

2. Firewall , adalah sebuah sistem yang dirancang untuk mencegah akses tidak sah ke atau dari jaringan pribadi. Firewall dapat diimplementasikan baik sebagai perangkat keras dan perangkat lunak atau kombinasi keduanya.

3. *Algoritma Message Digest dan Digital Signatures*

Sebuah tanda tangan digital terdiri dari dua potongan informasi: string bit yang dihitung dari data yang sedang "ditandatangani," bersama dengan kunci privat dari individu atau organisasi yang ingin tanda tangannya. Tanda tangan dapat digunakan untuk memverifikasi bahwa data berasal dari individu atau organisasi.

4. Digital Certificates

adalah lampiran ke sebuah pesan elektronik yang digunakan untuk tujuan keamanan, verifikasi pengguna akan mengirimkan sebuah pesan yang dia klaim, untuk penerima dengan menyediakan kodekan jawaban. standar paling banyak digunakan adalah sertifikat digital X.509.

5. Kerberos, Kerberos memiliki fungsi mirip dengan server sertifikat: untuk mengidentifikasi dan memvalidasi pengguna. Pentingnya Kerberos adalah bahwa ia menyediakan satu server keamanan terpusat untuk semua data dan sumber daya pada jaringan. Akses database, login, kontrol otorisasi, dan fitur keamanan lainnya yang terpusat di server

6. Secure Socket Layer dan Secure HTTP

Secure Socket Layer (SSL) yang dikembangkan oleh Netscape untuk transmisi dokumen pribadi melalui Internet. SSL bekerja dengan menggunakan sebuah kunci pribadi untuk mengenkripsi data yang ditransfer melalui sambungan SSL. Baik Firefox dan Internet Explorer mendukung SSL, dan banyak situs Web menggunakan protokol ini untuk mendapatkan informasi pengguna rahasia,

7. Transaksi Elektronik Aman dan Aman Teknologi

Transaksi Secure Electronic Transaction (SET) adalah protokol standar, terbuka interoperabel untuk pemrosesan cardtransactions kredit melalui Internet, yang diciptakan bersama oleh Netscape, Microsoft, Visa, Mastercard, GTE, SAIC, Terisa Sistem, dan VeriSign. SETS tujuannya adalah untuk memungkinkan transaksi kartu kredit menjadi sesederhana dan aman di Internet seperti yang di toko-toko ritel.

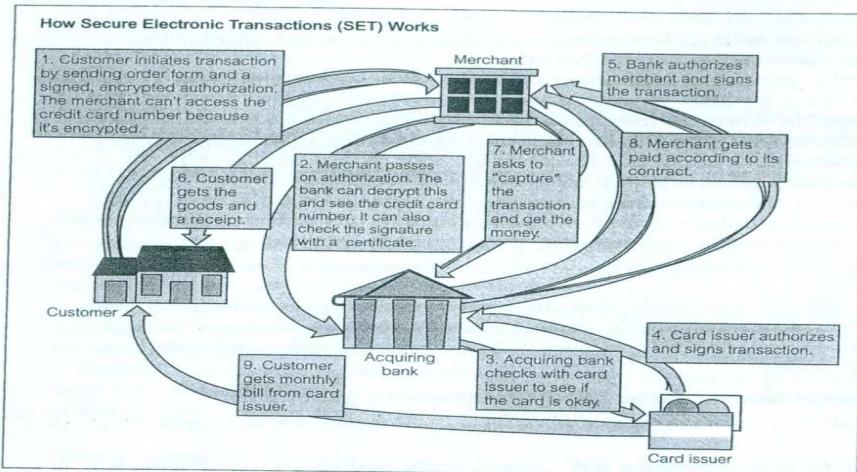


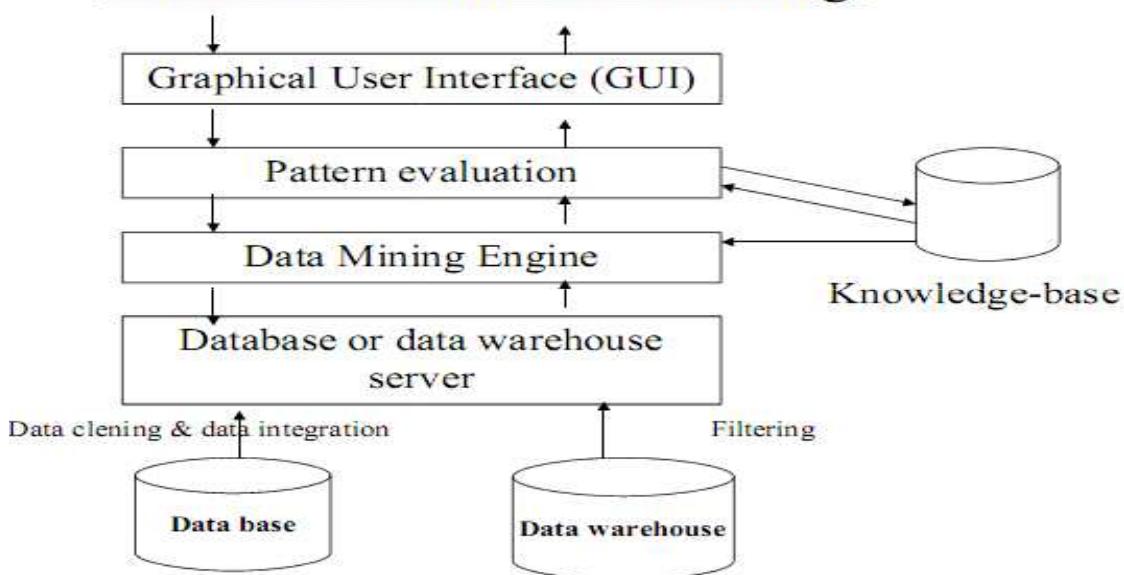
Figure 20.11 A SET transaction.

Gambar Transaksi Elektronik Aman dan Aman Teknologi

PERTEMUAN 13

ARSITEKTUR & MODEL DATA MINING

Arsitektur : Sistem Data Mining



28 September 2005

Arsitektur dan Model Data Mining

Keterangan :

1. Data cleaning (Pembersihan Data) : untuk membuang data yang tidak konsisten dan noise)
2. Data integration : penggabungan data dari beberapa sumber
3. Data Mining Engine : Mentransformasikan data menjadi bentuk yang sesuai untuk di mining
4. Pattern evaluation : untuk menemukan yang bernilai melalui knowledge base
5. Graphical User Interface (GUI) : untuk end user

Semua tahap bersifat interaktif di mana user terlibat langsung atau dengan perantaraan knowledge base

Model Data Mining

- Prediction Methods

Menggunakan beberapa variabel untuk memprediksi sesuatu atau suatu nilai yang akan datang.

- Description Methods

Mendapatkan pola penafsiran (humaninterpretable patterns) untuk menjelaskan data.

Data Mining

Prediktif

- Klasifikasi
- Decision tree
- Analisis Time series
- Regresi
- Prediksi
- Jaringan syaraf tiruan
- Data Mining

Deskriptif

- Klastering
- Summarization
- Aturan Asosiasi
- (Assosiation Rule)
- Sequence Discovery

Klasifikasi

- Proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data dengan tujuan untuk dapat memprediksi kelas dari suatu objek yang labelnya tidak diketahui
- Contoh : Mendeteksi Penipuan
- Tujuan : Memprediksi kasus kecurangan transaksi kartu kredit.
- Pendekatan :
 - Menggunakan transaksi kartu kredit dan informasi dilihat dari atribut account holder

- Kapan customer melakukan pembelian, Dengan cara apa customer membayar, seberapa sering customer membayar secara tepat waktu, dll
- Beri nama/tanda transaksi yang telah dilaksanakan sebagai transaksi yang curang atau yang baik. Ini sebagai atribut klass (the class attribute.)
- Pelajari model untuk class transaksi
- Gunakan model ini untuk mendeteksi kecurangan dengan meneliti transaksi kartu kredit pada account.

Regression

Digunakan untuk memetakan data dengan prediksi atribut bernilai real

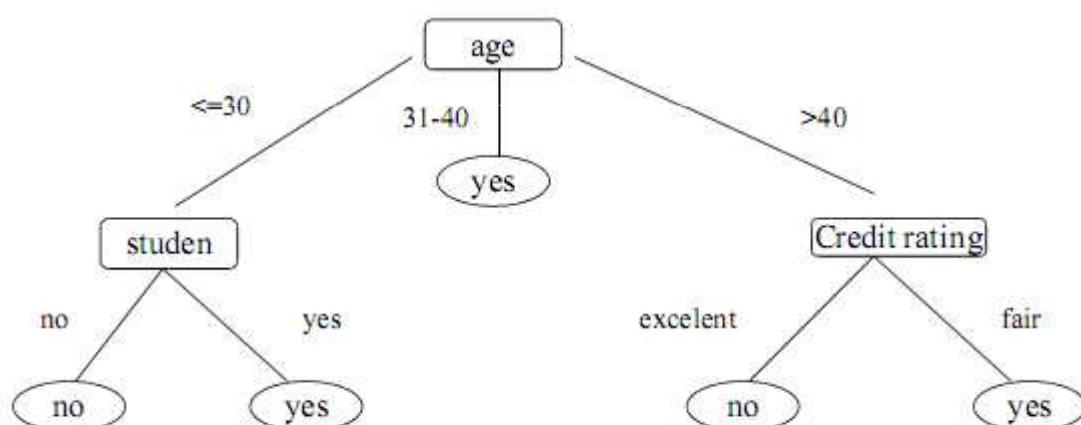
Contoh:

- Memprediksi jumlah penjualan produk baru pada advertising expenditure.
- Memprediksi kecepatan memutar (wind velocities) pada fungsi temperatur, tekanan udara , dll

Decision tree (Pohon keputusan)

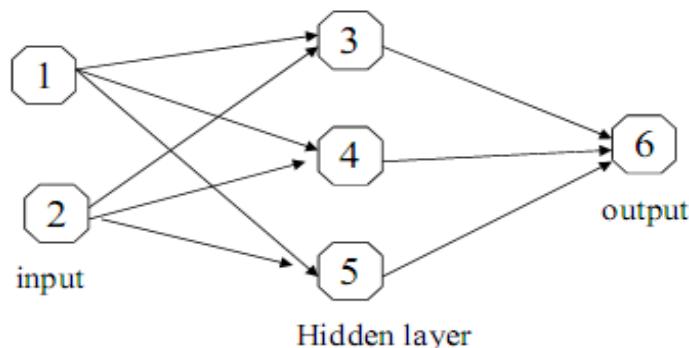
Salah satu model klasifikasi yang mudah di interpretasikan

Contoh : identifikasi pembeli komputer (dari decision tree di bawah ini ternyata salah satu kelompok yang potensial adalah orang yang berusia < 30 dan pelajar



Neural Network (Jaringan syaraf tiruan)

- Jaringan syaraf buatan di mulai dengan layer input, dimana tiap simpul berkorespondensi dengan variabel prediktor.
- Simpul-simpul input ini terhubung ke beberapa simpul dalam hidden layer. Dan simpul dalam hidden layer dapat terhubung ke simpul lain dalam hidden layer atau ke output layer.
- Output layer terdiri dari satu atau beberapa variable respon



- **Telekomunikasi**

Data mining digunakan untuk melihat jutaan transaksi yang masuk dengan tujuan menambah layanan otomatis

- **Keuangan**

Data mining digunakan untuk mendeteksi transaksi keuangan yang mencurigakan dimana akan susah dilakukan jika menggunakan analisis standar.

- **Asuransi**

Australian Health Insurance Commission menggunakan data mining untuk mengidentifikasi layanan kesehatan dan berhasil menghemat satu juta dollar pertahun

- Olah raga

IBM Advanced Scout menggunakan data mining untuk menganalisis statistik permainan NBA dalam rangka competitive advantage untuk tim New York Knicks

- Astronomi

Jet Propulsion Laboratory (JPL) di Pasadena dan Pulomar Observatory menemukan 22 quasar dengan bantuan data mining.

- Internet Web Surf-Aid

IBM Surf-Aid menggunakan algoritma data mining untuk mendata akses halaman Web khususnya berkaitan dengan pemasaran melalui web.

Tools Data Mining

- Karakteristik-karakteristik penting dari tool data mining meliputi :
 - Data preparation facilities
 - Selection of data mining operation (algorithms)
 - Product scalability and performance
 - Facilities for visualization of result
- Data mining tool, meliputi :
 - Integral Solution Ltd's Clementine
 - DataMind Corp's Data Crusher
 - IBM's Intelligent Miner
 - Silicon Graphics Inc.'s MineSet
 - Informations Discovery Inc.'s Data Mining Suite
 - SAS Institute Inc.'s SAS System and Right Information System'Thought.

Evolusi Database

- Th 1960
 - Pengumpulan data, pembuatan data, IMS dan network DBMS
- Th 1970
 - Model data relasional, Implementasi DBMS relasional
- Th 1980
 - RDBMS, Model data lanjutan (extended-relational, OO, deductive)
- Th 1990
 - Data mining, data warehouse, database multimedia, dan Web database.
- Th 2000
 - Stream data managemen dan mining – Data mining dengan berbagai variasi aplikasi – Teknologi web dan sistem informasi global

Teknik – teknik Database

Searching

- Searching dilakukan untuk memeriksa serangkaian item yang memiliki sifatsifat yang diinginkan.
- Tindakan untuk menemukan suatu item tertentu baik yang diketahui keberadaannya maupun tidak.
- Memasukkan kata dalam suatu program komputer untuk membandingkan dengan informasi yang ada dalam database.

Indexing

- Indexing adalah struktur-struktur akses yang digunakan untuk mempercepat respon dalam mendapatkan record-record pada kondisi-kondisi pencarian tertentu.
- Indexing field adalah suatu struktur akses index yang biasanya menjelaskan field tunggal dari suatu file.
- Indexing organization memberikan efisiensi akses ke record-record secara berurut atau random.

Data Reduction

- *Data reduction adalah transformasi suatu masalah ke masalah lain dan dapat digunakan untuk mendefinisikan serangkaian masalah yang kompleks.*
- *Data reduction merupakan teknik yang digunakan untuk mentransformasi dari data mentah ke bentuk format data yang lebih berguna. Sebagai contoh grouping, summing dan averaging data.*
- *Data reduction dilakukan untuk mengatasi ukuran data yang terlalu besar. Ukuran data yang terlalu besar dapat menimbulkan ketidakefisienan proses dan peningkatan biaya pemrosesan.*
- *Data reduction dilakukan dalam tahap data preprocessing pada rangkaian proses Knowledge Discovery Databases (KDD) sebelum data mining dengan tujuan mengurangi ukuran data yang besar.*

OLAP (On-line analytical processing)

- OLAP adalah suatu sistem atau teknologi yang dirancang untuk mendukung proses analisis kompleks dalam rangka mengungkapkan kecenderungan pasar dan faktor-faktor penting dalam bisnis
- OLAP ditandai dengan kemampuannya menaikkan atau menurunkan dimensi data sehingga kita dapat menggali data sampai pada level yang sangat detail dan memperoleh pandangan yang lebih luas mengenai objek yang sedang kita analisis.
- OLAP secara khusus memfokuskan pada pembuatan data agar dapat diakses pada saat pendefinisiannya kembali dimensi.
- OLAP dapat digunakan membuat rangkuman dari multidimensi data yang berbeda, rangkuman baru dan mendapatkan respon secara online, dan memberikan view dua dimensi pada data cube multidimensi secara interaktif.

OLAP (ONLINE ANALYTICAL PROCESSING)

Aplikasi OLAP didominasi oleh ad hoc, query kompleks. Dalam istilah SQL, ini adalah query yang melibatkan kelompok-oleh dan operator agregasi. Cara alami untuk berpikir tentang query OLAP adalah dalam hal model data multidimensi. Kita mulai bagian ini dengan menyajikan model data multidimensi dan membandingkannya dengan representasi data relasional.

MODEL DATA MULTIDIMENSIONAL

- Dalam model data multidimensi, fokusnya adalah pada koleksi **langkah-langkah** numerik. Setiap ukuran tergantung pada set **dimensi**.
- Beberapa sistem OLAP, misalnya, Essbase dari Software Arbor, sebenarnya menyimpan data dalam array multidimensi. Sistem OLAP yang menggunakan array untuk menyimpan dataset multidimensi disebut **OLAP multidimensi (MOLAP)** sistem.

- Operasi yang didukung oleh model ini sangat dipengaruhi oleh alat pengguna akhir seperti spreadsheet. Tujuannya adalah untuk memberikan pengguna akhir yang bukan ahli SQL antarmuka yang intuitif dan kuat untuk umum tugas analisis businessoriented. Pengguna diharapkan untuk menimbulkan ad hoc query secara langsung, tanpa bergantung pada programmer aplikasi database.

PERTEMUAN 14

DATA WAREHOUSE

Data Warehouse

Definisi :

- Data Warehouse adalah Pusat repositori informasi yang mampu memberikan database berorientasi subyek untuk informasi yang bersifat historis yang mendukung DSS (Decision Support System) dan EIS (Executive Information System).
- Salinan dari transaksi data yang terstruktur secara spesifik pada query dan analisa.
- Salinan dari transaksi data yang terstruktur spesifik untuk query dan laporan

Tujuan :

- Meningkatkan kualitas dan akurasi informasi bisnis dan mengirimkan informasi ke pemakai dalam bentuk yang dimengerti dan dapat diakses dengan mudah.

4 KARAKTERISTIK DATA WAREHOUSE

1. Subject oriented

- Data yang disusun menurut subyek berisi hanya informasi yang penting bagi pemrosesan decision support.
- Data yang disusun menurut subyek berisi hanya informasi yang penting bagi pemrosesan decision support.
- Database yang semua informasi yang tersimpan dikelompokkan berdasarkan subyek tertentu misalnya : pelanggan, gudang, pasar, dsb.
- Semua informasi tersebut disimpan dalam suatu system data warehouse.
- Data setiap subyek dirangkum ke dalam dimensi, misalnya: periode waktu, produk, wilayah, dsb, sehingga dapat memberikan nilai sejarah untuk bahan analisa.

2. Integrated

Jika data terletak pada berbagai aplikasi yang terpisah dalam suatu lingkungan operasional, encoding data sering tidak seragam sehingga bila data dipindahkan ke data warehouse maka coding akan diasumsikan sama seperti lazimnya.

3. Time-variant

Data warehouse adalah tempat untuk storing data selama 5 sampai 10 tahun atau lebih, data digunakan untuk perbandingan atau perkiraan dan data ini tidak dapat diperbarui.

4. Non volatile

Data tidak dapat diperbarui atau dirubah tetapi hanya dapat ditambah dan dilihat.

MASALAH-MASALAH DALAM MENERAPKAN DATA WAREHOUSE

- Dokumentasi dan pengelolaan metadata dari data warehouse.
- Penentuan aturan dalam proses transformasi untuk menetapkan berbagai sumber legacy data yang akan dimasukkan ke dalam data warehouse.
- Pencapaian proses pengembangan yang handal, baik dalam membangun, mengimplementasikan, maupun memelihara data warehouse.

KEUNTUNGAN DATAWAREHOUSE

- Datawarehouse menyediakan model data yang bervariasi, dan tidak bergantung pada satu sumber data saja. Hal ini memudahkan pimpinan perusahaan/manager membuat laporan dan menganalisa.
- Saat me-load data ke dalam datawarehouse, data yang tidak konsisten akan diketahui dan secepatnya dirubah. Mendukung proses pembuatan laporan, agar keputusan yang diambil adalah keputusan yang benar sesuai data.
- Keamanan informasi didalam datawarehouse terjamin, karena datawarehouse selalu digunakan dan dimonitor oleh pengguna datawarehouse tersebut.
- Dalam membuat laporan tidak membuat proses transaksi yang ada menjadi lambat, karena datawarehouse terpisah dengan database operasional.
- Datawarehouse menyediakan berbagai macam bentuk laporan yang terbaru.

KERUGIAN DATAWAREHOUSE

- Datawarehouse tidak cocok untuk data yang tidak struktur.
- Data perlu di extract, diubah, dan di load ke datawarehouse, sehingga membutuhkan waktu (**delay**) kerja untuk datawarehouse yang belum terbentuk.
- Semakin lama masa hidup bisnis yang menggunakan datawarehouse, maka semakin banyak biaya yang dikeluarkan oleh perusahaan untuk memodifikasi teknologi datawarehouse atau perawatan berjalan datawarehouse.
- Jika data yang diambil lambat, maka data yang dimiliki di datawarehouse tidak berkualitas/ sehingga laporan tidak optimal.

Ada banyak tantangan dalam menciptakan dan memelihara sebuah data warehouse yang besar. Sebuah skema database yang baik harus dirancang untuk menahan koleksi terpadu dari data yang disalin dari berbagai sumber. Sebagai contoh, sebuah warehouse perusahaan mungkin termasuk persediaan dan database personil departemen ', bersama-sama dengan database penjualan dikelola oleh kantor di negara yang berbeda. Karena source database sering dibuat dan dipelihara oleh kelompok yang berbeda, ada sejumlah ketidaksesuaian semantik di database ini, seperti unit mata uang yang berbeda, nama yang berbeda untuk atribut yang sama, dan perbedaan dalam bagaimana tabel dinormalisasi atau terstruktur; perbedaan-perbedaan ini harus didamaikan ketika data dibawa ke warehouse. Setelah skema warehouse dirancang, warehouse harus diisi, dan dari waktu ke waktu, itu harus tetap konsisten dengan database sumber.

Ciri-ciri Data Warehouse

Terdapat 4 karakteristik data warehouse

1. Subject oriented

- Data yang disusun menurut subyek berisi hanya informasi yang penting bagi pemrosesan decision support.
- Database yang semua informasi yang tersimpan di kelompokkan berdasarkan subyek tertentu misalnya: pelanggan, gudang, pasar, dsb.
- Semua Informasi tersebut disimpan dalam suatu sistem *data warehouse*.
- *Data-data di setiap subyek dirangkum ke dalam dimensi, misalnya : periode waktu, produk, wilayah, dsb, sehingga dapat memberikan nilai sejarah untuk bahan analisa.*

Ciri-ciri Data Warehouse

2. Integrated

- Jika data terletak pada berbagai aplikasi yang terpisah dalam suatu lingkungan operasional, encoding data sering tidak seragam sehingga bila data dipindahkan ke data warehouse maka coding akan diasumsikan sama seperti lazimnya.

3. Time-variant

- Data warehouse adalah tempat untuk storing data selama 5 sampai 10 tahun atau lebih, data digunakan untuk perbandingan atau perkiraan dan data ini tidak dapat diperbarui.

4. Non volatile

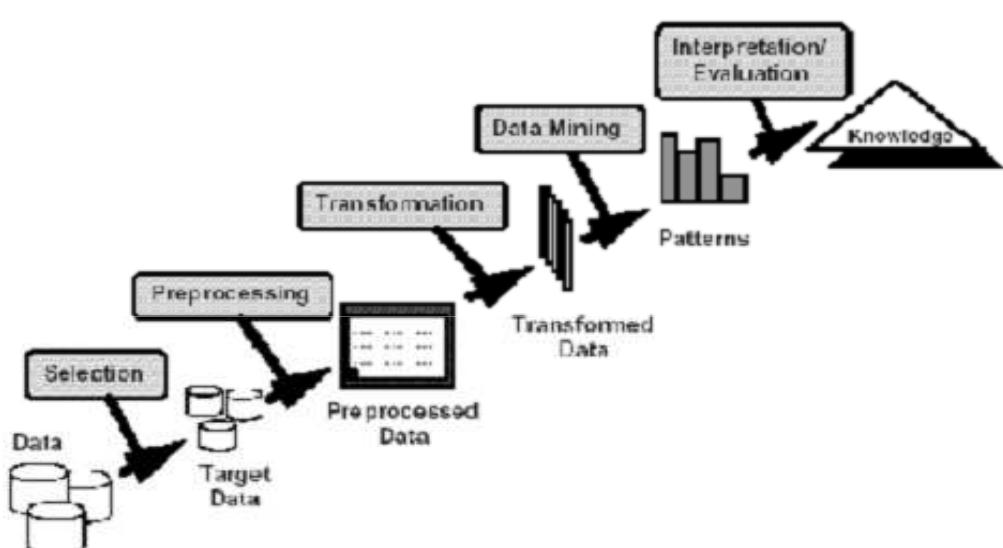
- Data tidak dapat diperbarui atau dirubah tetapi hanya dapat ditambah dan dilihat.

Data Preprocessing

- Data preprocessing menerangkan tipe-tipe proses yang melaksanakan data mentah untuk mempersiapkan proses prosedur yang lainnya.
- Dalam data mining menrasformasi data ke suatu format yang prosesnya lebih mudah dan efektif untuk kebutuhan pemakai, contohnya Neural Network.
- Terdapat beberapa alat dan metode yang berbeda yang digunakan untuk preprocessing seperti :
 - *Sampling* : menyeleksi subset representatif dari populasi data yang besar.
 - *Transformation* : memanipulasi data mentah untuk menghasilkan input tunggal.
 - *Denoising* : menghilangkan noise dari data
 - *Normalization* : mengorganisasi data untuk pengaksesan yang lebih spesifik
 - *Feature extraction* : membuka spesifikasi data yang signifikan dalam konteks tertentu.

KNOWLEDGE DISCOVERY IN DATABASE (KDD)

- KDD berhubungan dengan teknik integrasi dan penemuan ilmiah, interpretasi dan visualisasi dari pola-pola sejumlah kumpulan data.
- *Knowledge discovery in databases (KDD) adalah keseluruhan proses non-trivial untuk mencari dan mengidentifikasi pola (pattern) dalam data, dimana pola yang ditemukan bersifat sah, baru, dapat bermanfaat dan dapat dimengerti.*



Gambar. 1. Tahapan KDD

1. Data Selection

- Menciptakan himpunan data target , pemilihan himpunan data, atau memfokuskan pada subset variabel atau sampel data, dimana penemuan (discovery) akan dilakukan.
- Pemilihan (seleksi) data dari sekumpulan data operasional perlu dilakukan sebelum tahap penggalian informasi dalam KDD dimulai. Data hasil seleksi yang akan digunakan untuk proses *data mining*, *disimpan dalam suatu berkas, terpisah dari basis data operasional*.

2. Pre-processing/ Cleaning

- Pemrosesan pendahuluan dan pembersihan data merupakan operasi dasar seperti penghapusan noise dilakukan.
- Sebelum proses *data mining* dapat dilaksanakan, perlu dilakukan proses *cleaning* pada data yang menjadi fokus KDD.
- Proses *cleaning* mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data, seperti kesalahan cetak (tipografi).
- Dilakukan proses enrichment, yaitu proses “memperkaya” data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan untuk KDD, seperti data atau informasi eksternal.

3. Transformation

- Pencarian fitur-fitur yang berguna untuk mempresentasikan data bergantung kepada goal yang ingin dicapai.
- Merupakan proses transformasi pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining*. *Proses ini merupakan proses kreatif dan sangat tergantung pada jenis atau pola informasi yang akan dicari dalam basis data*

4. Data mining

- Pemilihan tugas data mining; pemilihan goal dari proses KDD misalnya klasifikasi, regresi, clustering, dll.
- Pemilihan algoritma data mining untuk pencarian (searching)
- *Proses Data mining yaitu proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik, metode, atau algoritma dalam data mining sangat bervariasi. Pemilihan metode atau algoritma yang tepat sangat bergantung pada tujuan dan proses KDD secara keseluruhan.*

5. Interpretation/ Evaluation

- Penerjemahan pola-pola yang dihasilkan dari *data mining*.
- Pola informasi yang dihasilkan dari proses *data mining* perlu *ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan*.
- Tahap ini merupakan bagian dari proses KDD yang mencakup pemeriksaan apakah pola atau informasi yang ditemukan bertentangan dengan fakta atau hipotesa yang ada sebelumnya.