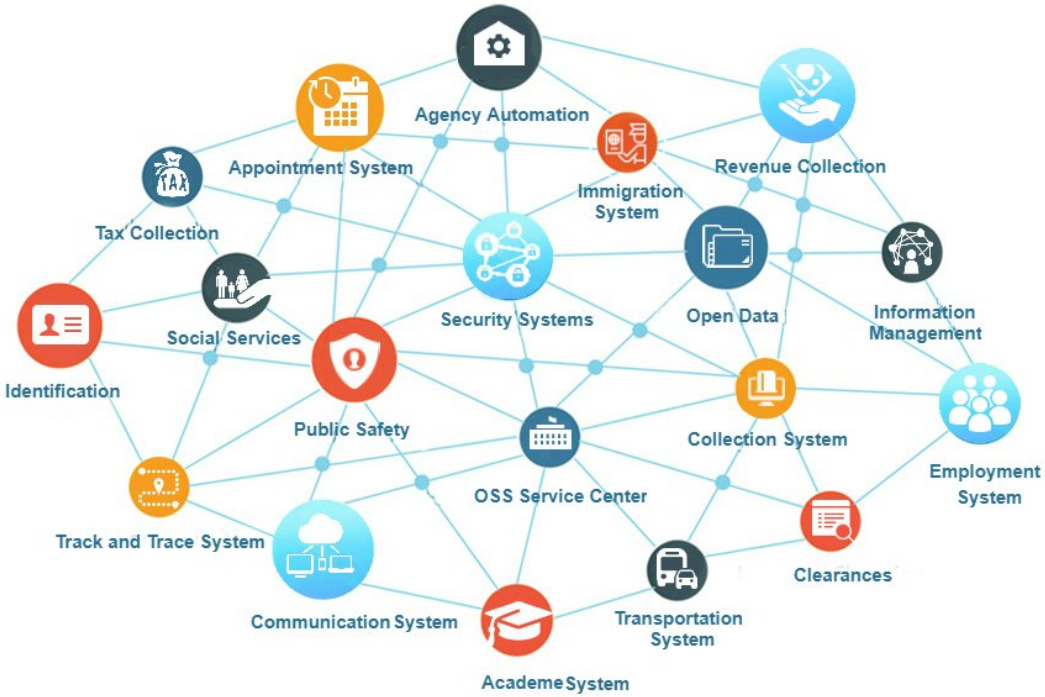


Permodelan Informasi Data

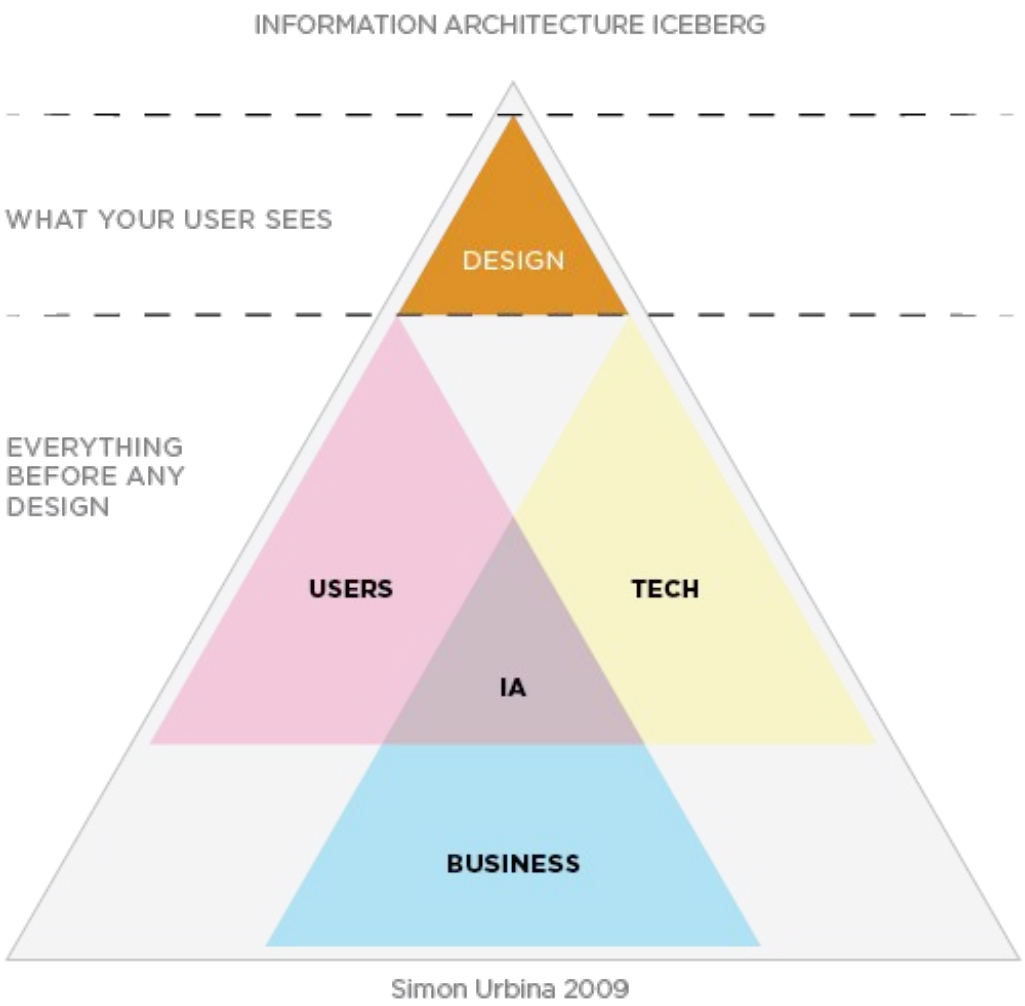
Dr. rer. nat. I Made Wiryana, SKom, SSi, MAppSc

Jadwal Pelatihan

- Hari 1
  - Arsitektur Bisnis
  - Bisnis Proses
  - Business Process Modelling Notation (BPMN)
  - Camunda Modeller dan menggambar Bisnis Proses
- Hari 2: Arsitektur Informasi dan Data
  - Information modelling dan IDEF1
  - Data modelling IDEF1x
  - Database modelling ERD, Schema, Docmodel
  - Metadata, Kamus Data
  -

Arsitektur Informasi

- Pengembang tidak boleh “egois”
- Memahami pengguna dan organisasi
- Memahami teknologi yang mungkin diimplementasikan



Konsepsi Alur Kerja

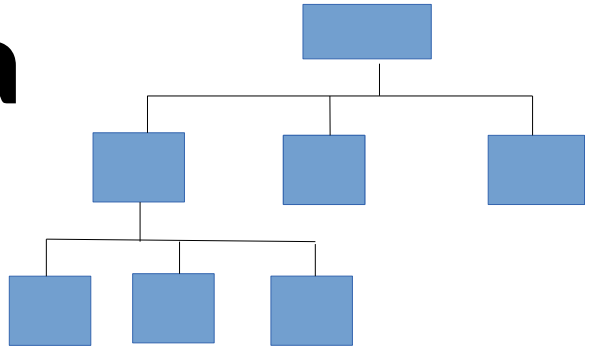
Arsitektur Business

Arsitektur Informasi
Arsitektur Data

Arsitektur Aplikasi

Arsitektur Teknologi

Tahapan pekerjaan



Struktur organisasi

Pencocokan Model Informasi dan Data

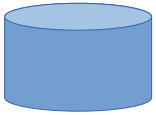
Survei Informasi

Survei Data

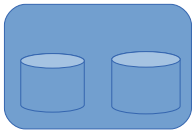
Survei SIM

Formulir

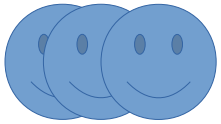
Laporan



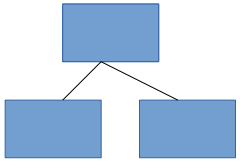
Data Elektronis



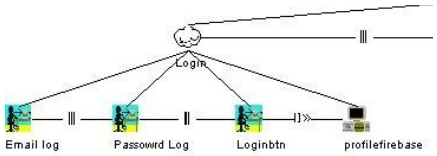
Sistem Informasi



User Model

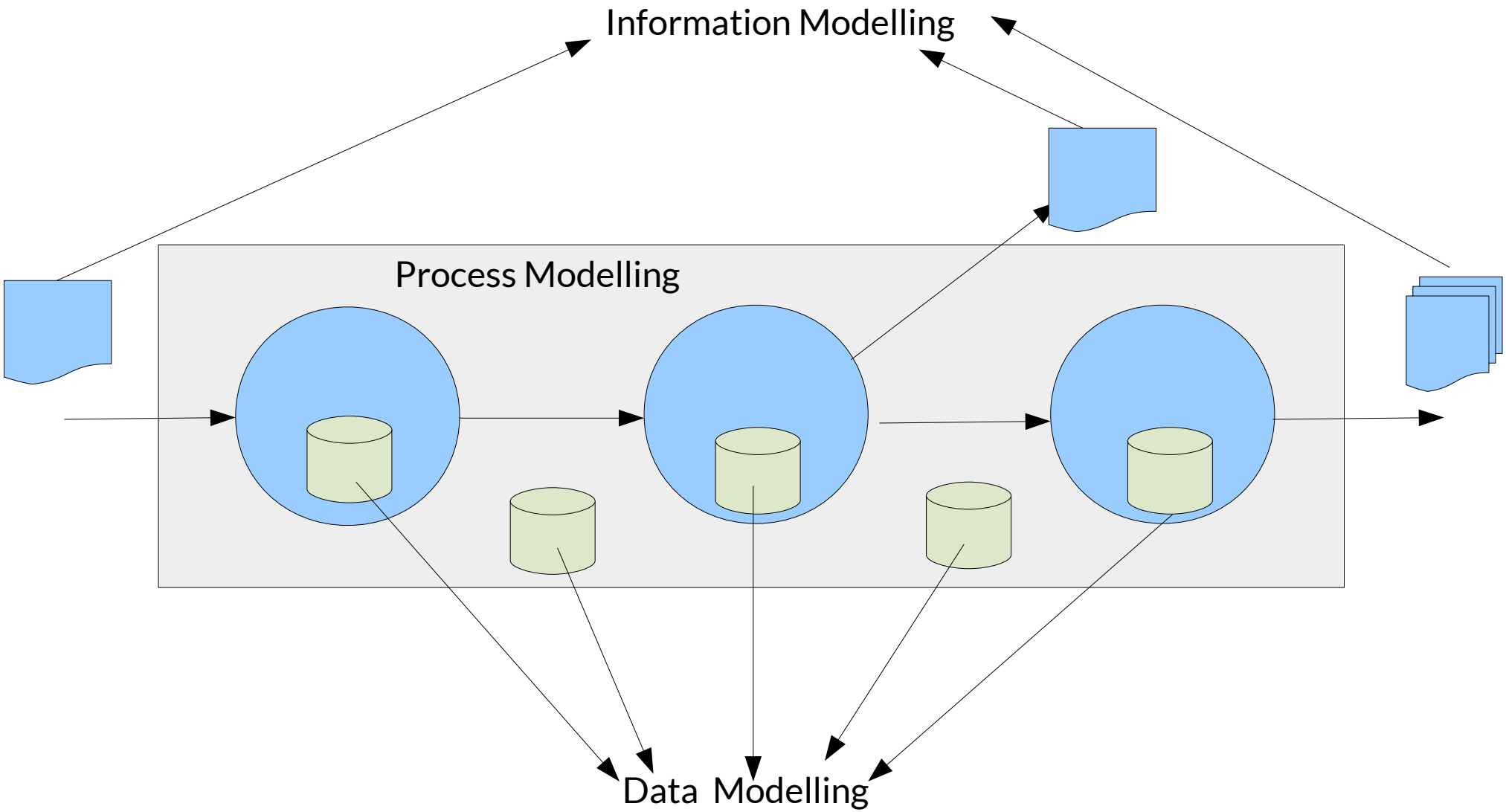


Task Model

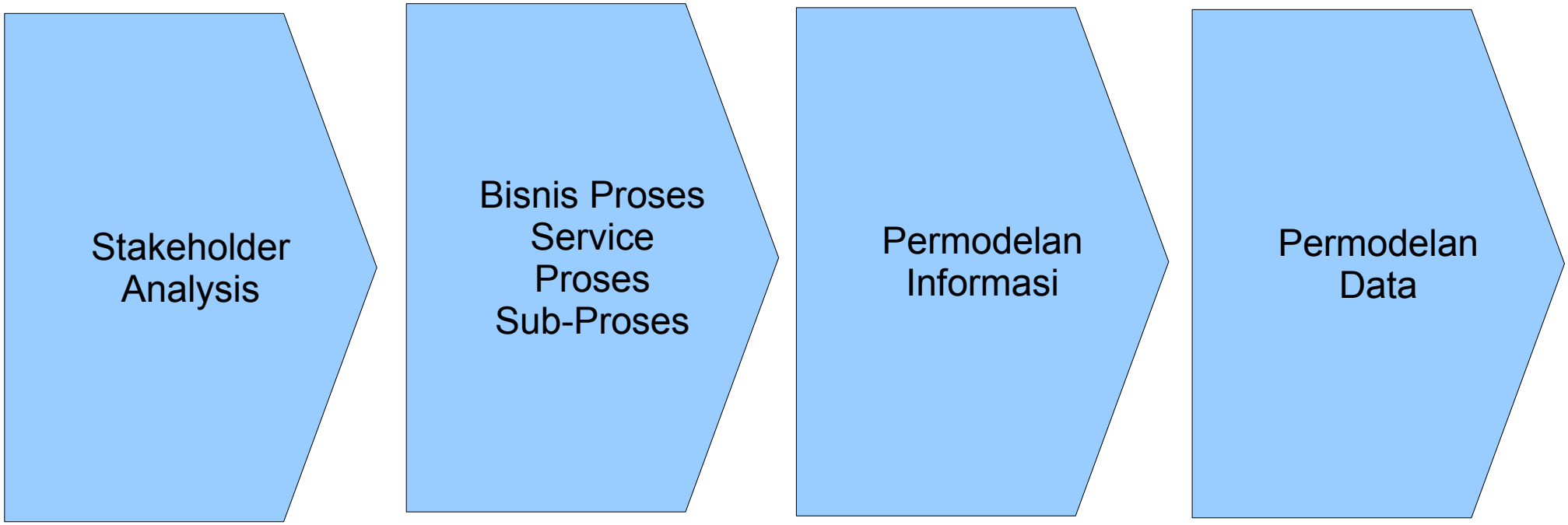


Interaction Model

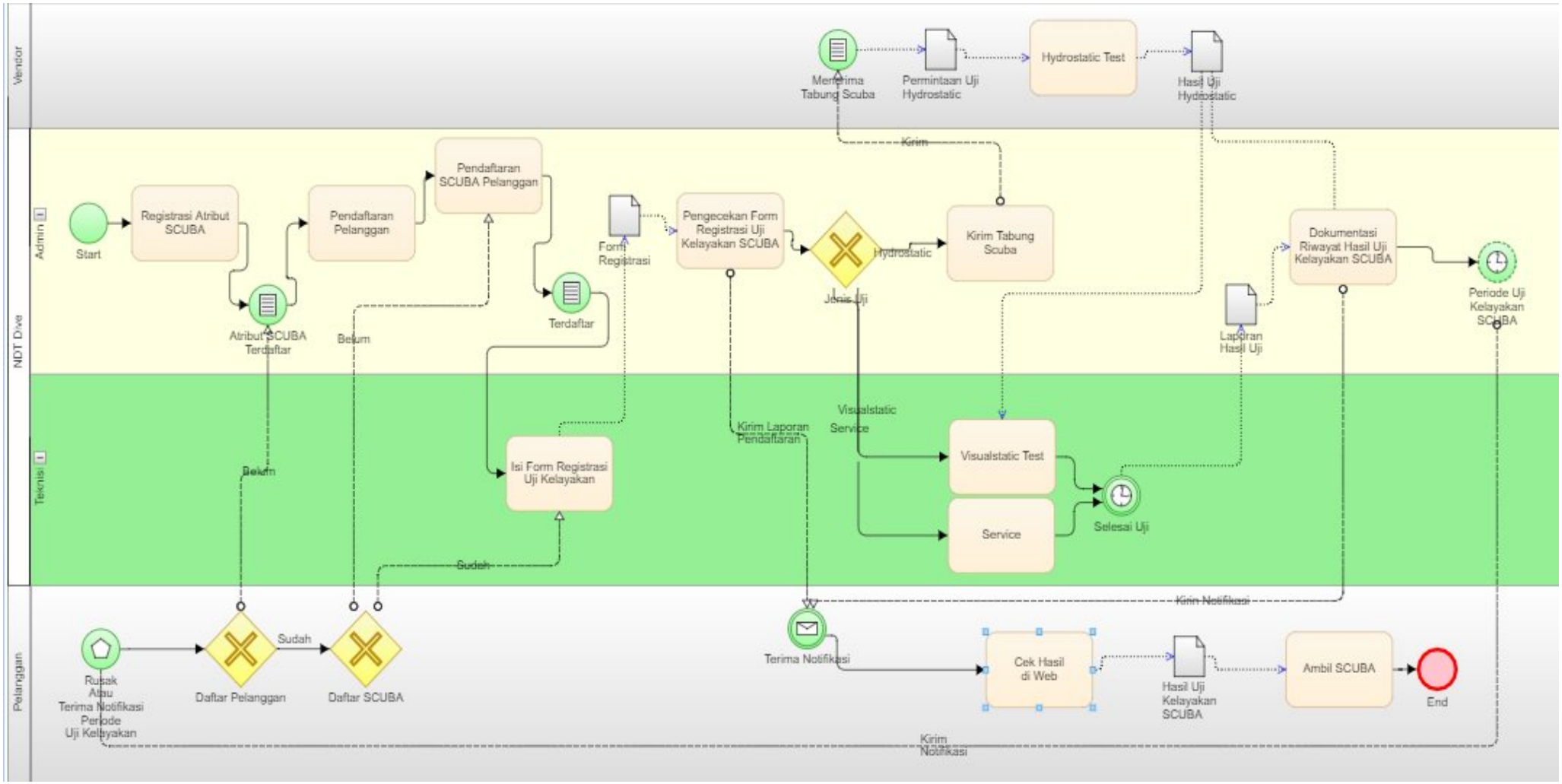
Tahapan

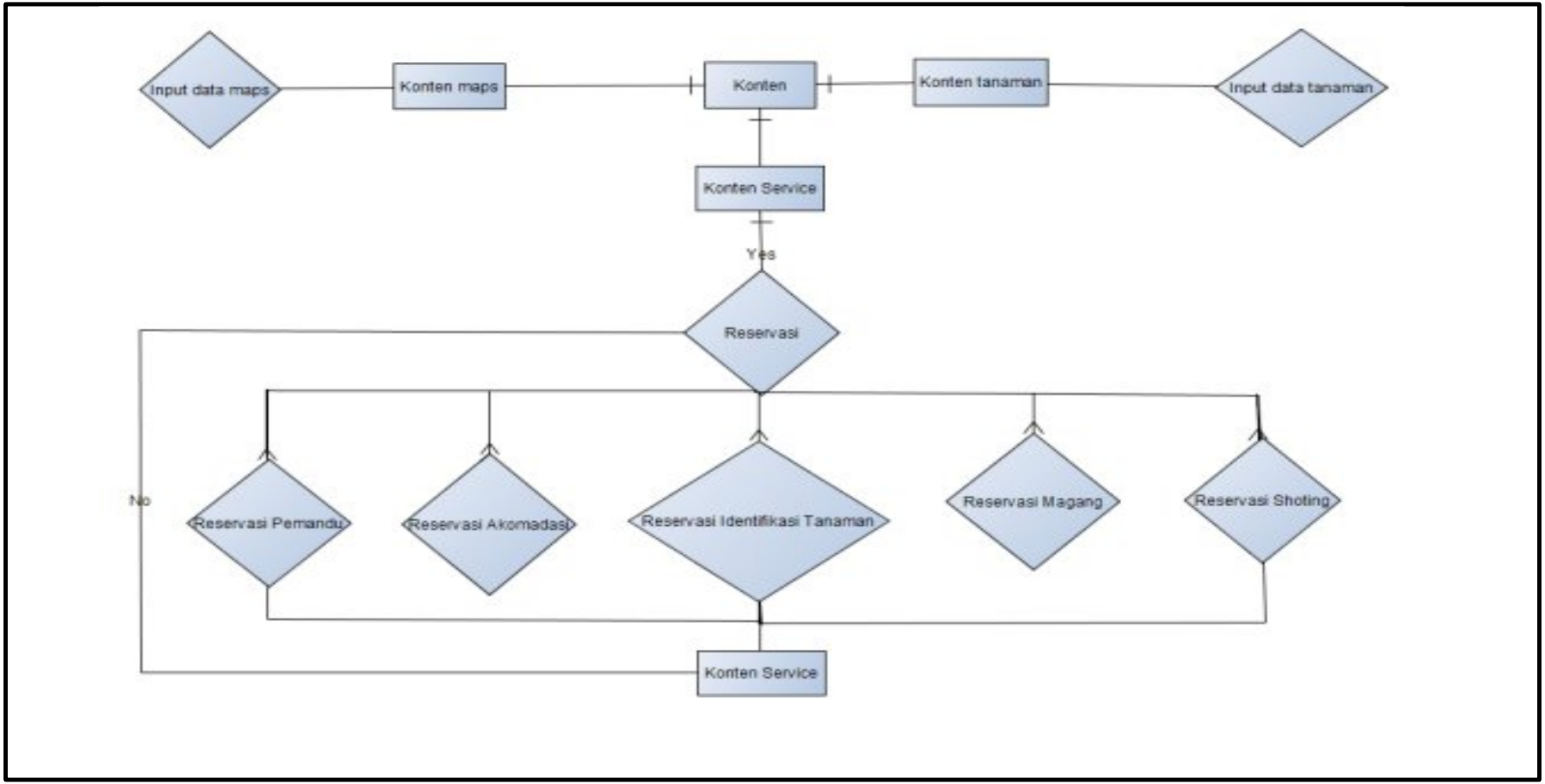


Tahapan


Stakeholder Analysis
Bisnis Proses
Service
Proses
Sub-Proses
Permodelan Informasi
Permodelan Data

Rekayasa *Workflow*

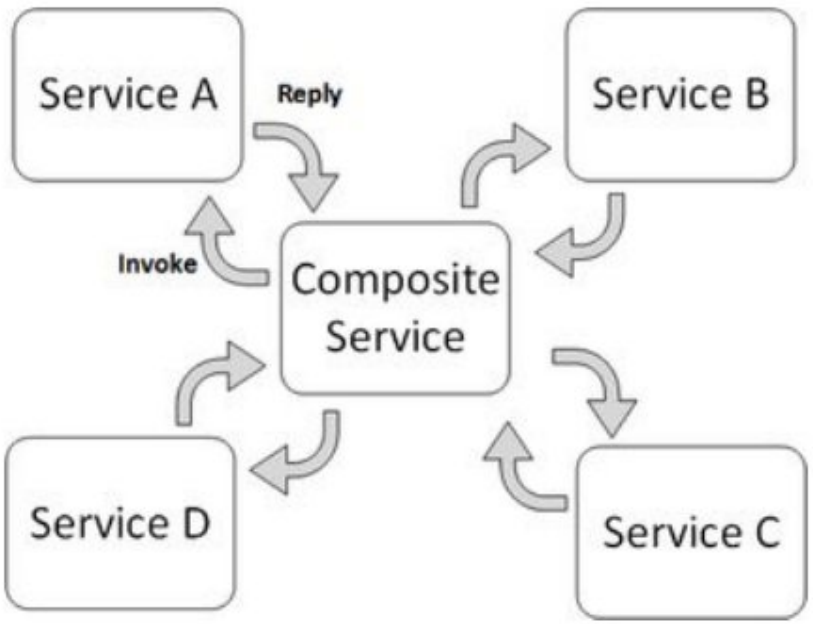


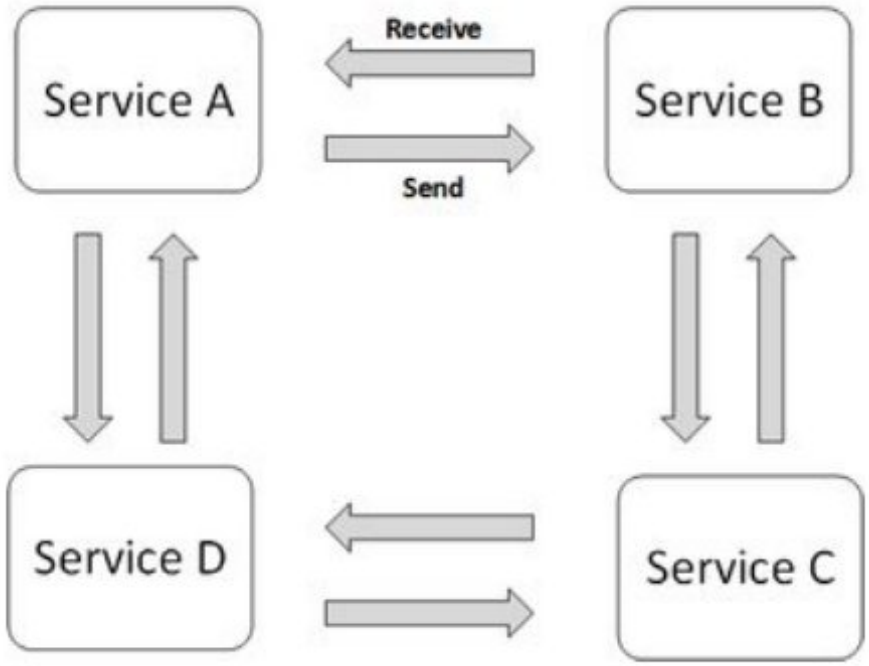


- SMOF adalah pengembangan metodologi proses berorientasi layanan bisnis
- Memiliki layanan konseptual, analisis, desain dan solusi.

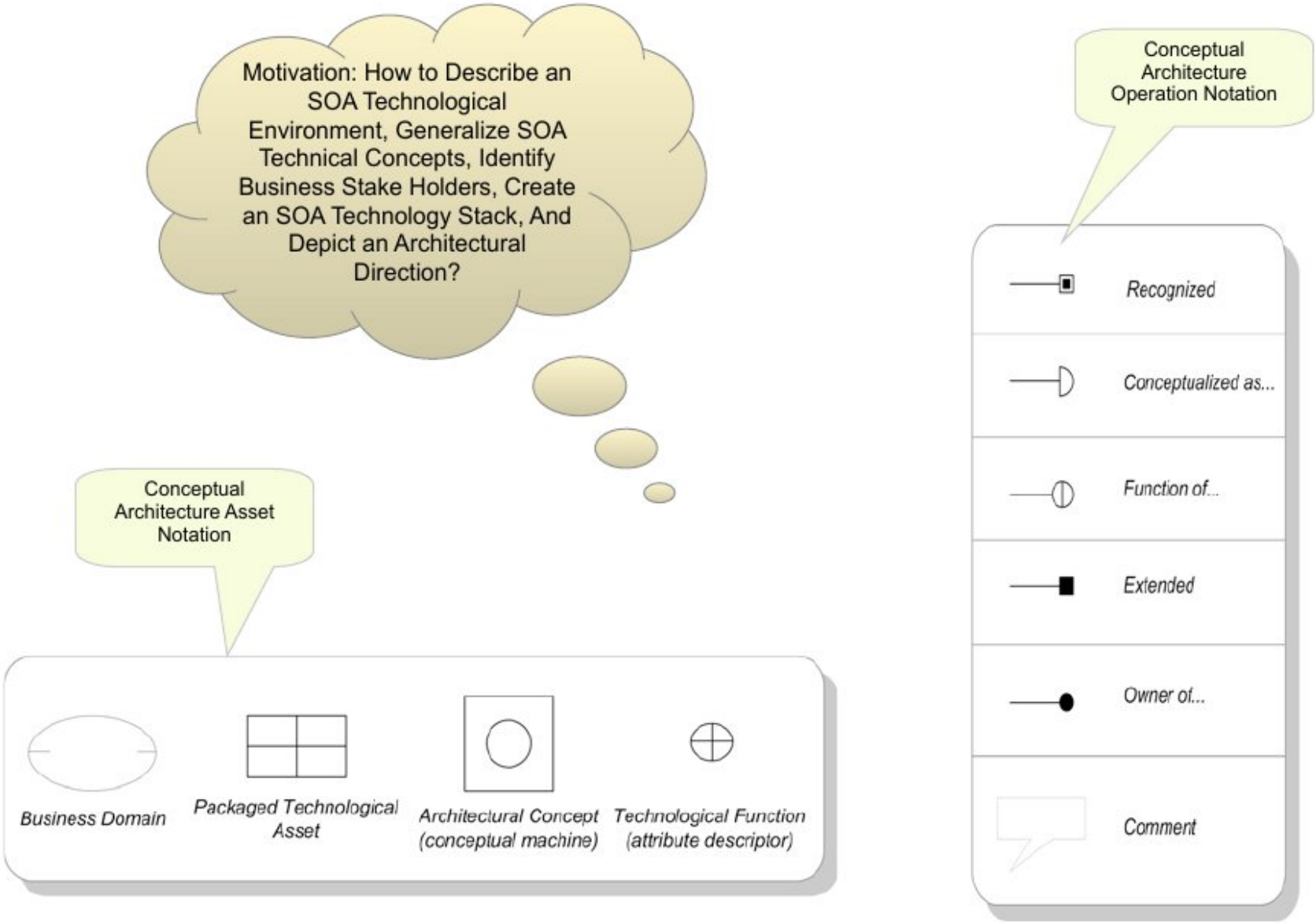
Integrasi layanan

- **Orkestrasi layanan** merepresentasikan layanan yang dikoordinasikan secara terpusat oleh sebuah koordinator yang disebut sebagai orkestrator, yang bertanggung jawab terhadap permintaan dan penggabungan layanan.
- **Koreografi layanan** adalah deskripsi umum dari layanan yang berpartisipasi yang didefisinikan dengan adanya pertukaran pesan, aturan interaksi dan perjanjian diantara dua atau lebih layanan. Koreografi sebagai pendekatan desentralisasi untuk komposisi layanan.

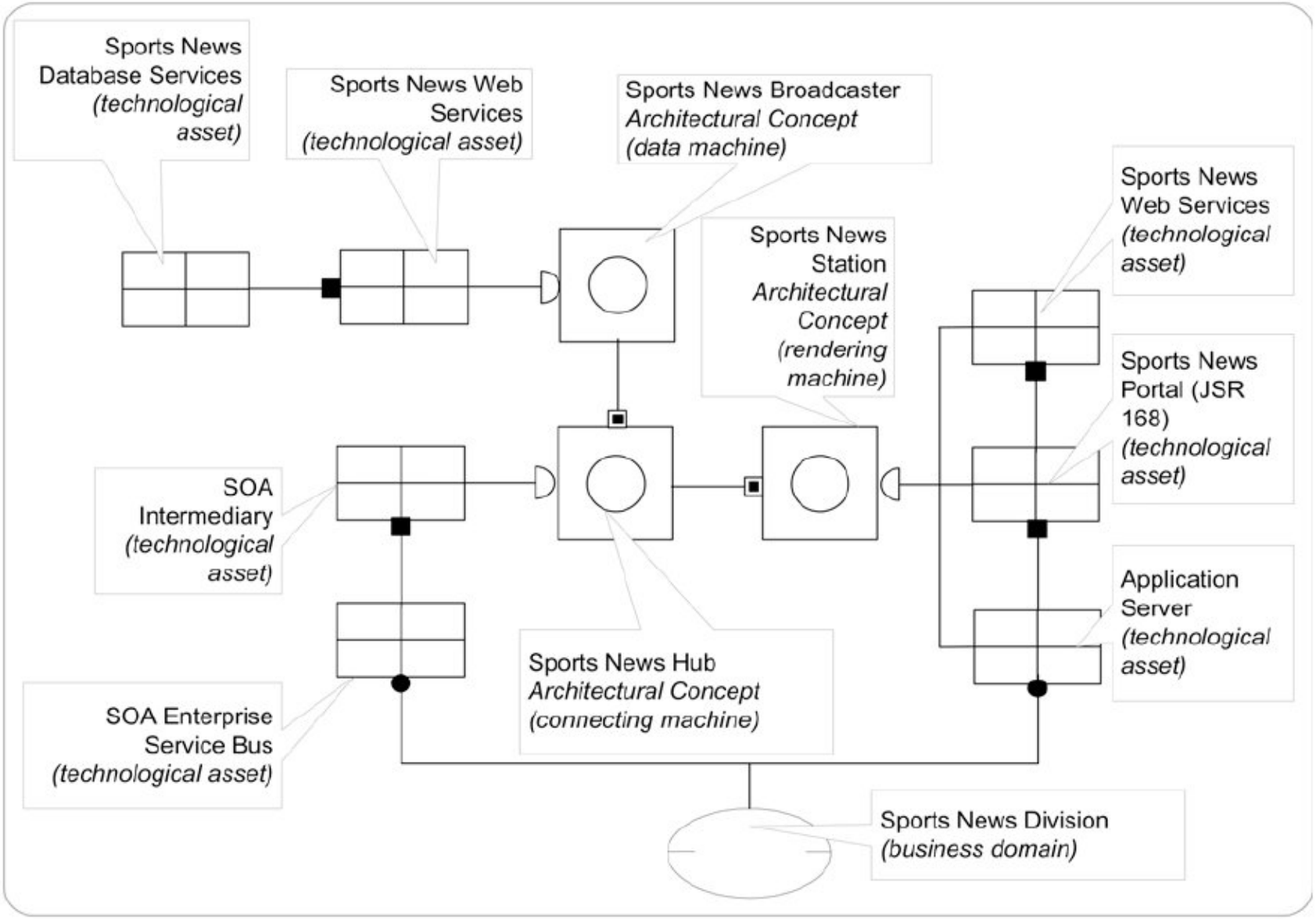




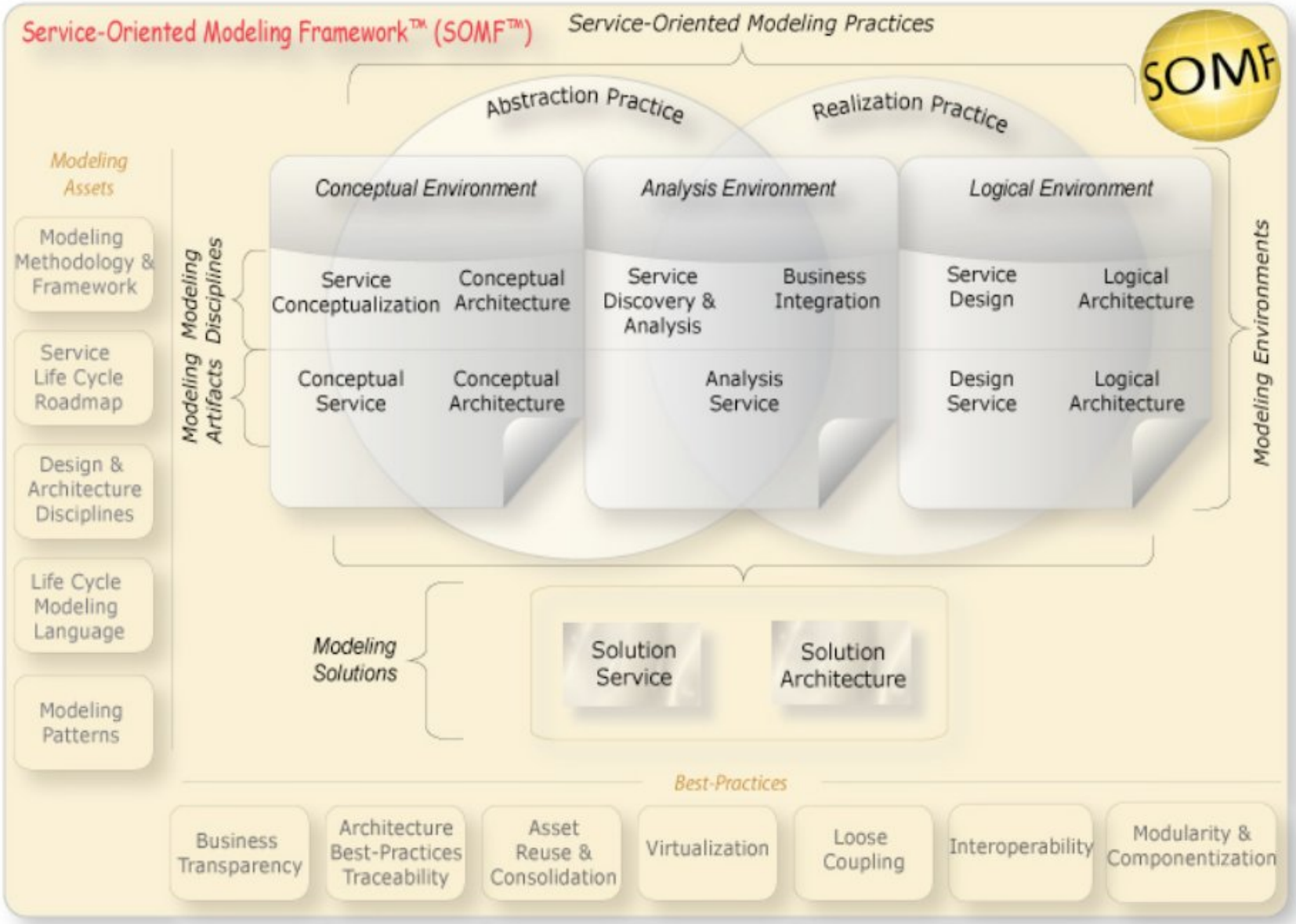
Service Oriented Conceptual Arch.



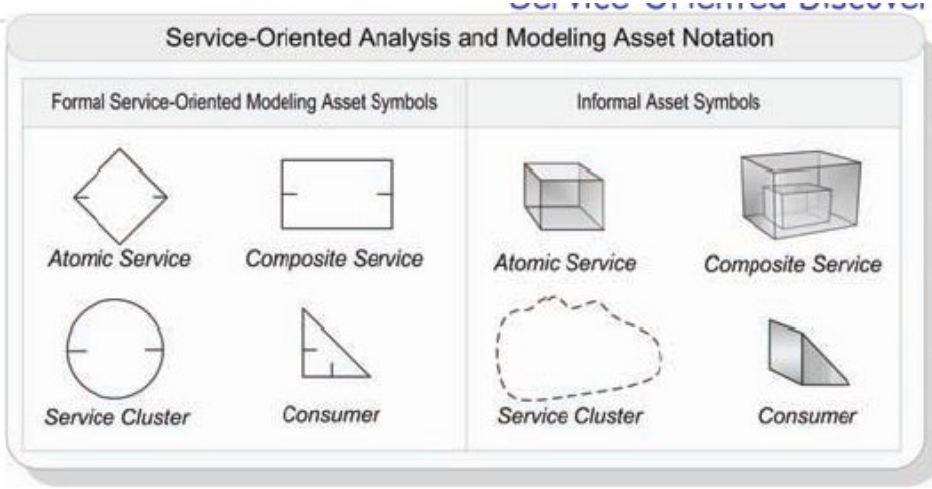
Service Oriented Conceptual Arch.

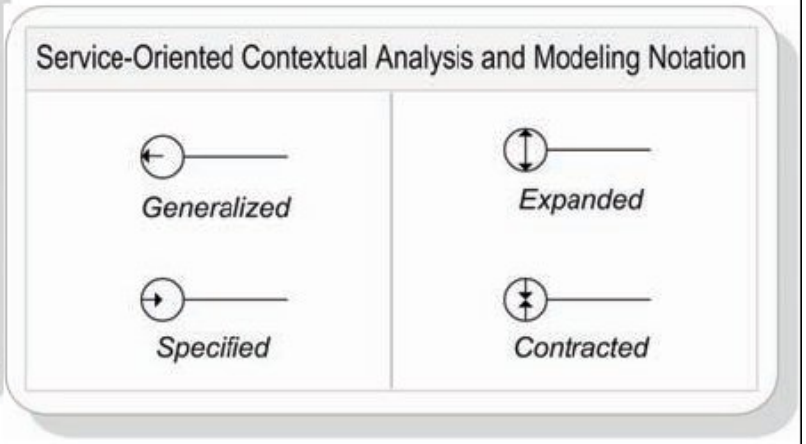


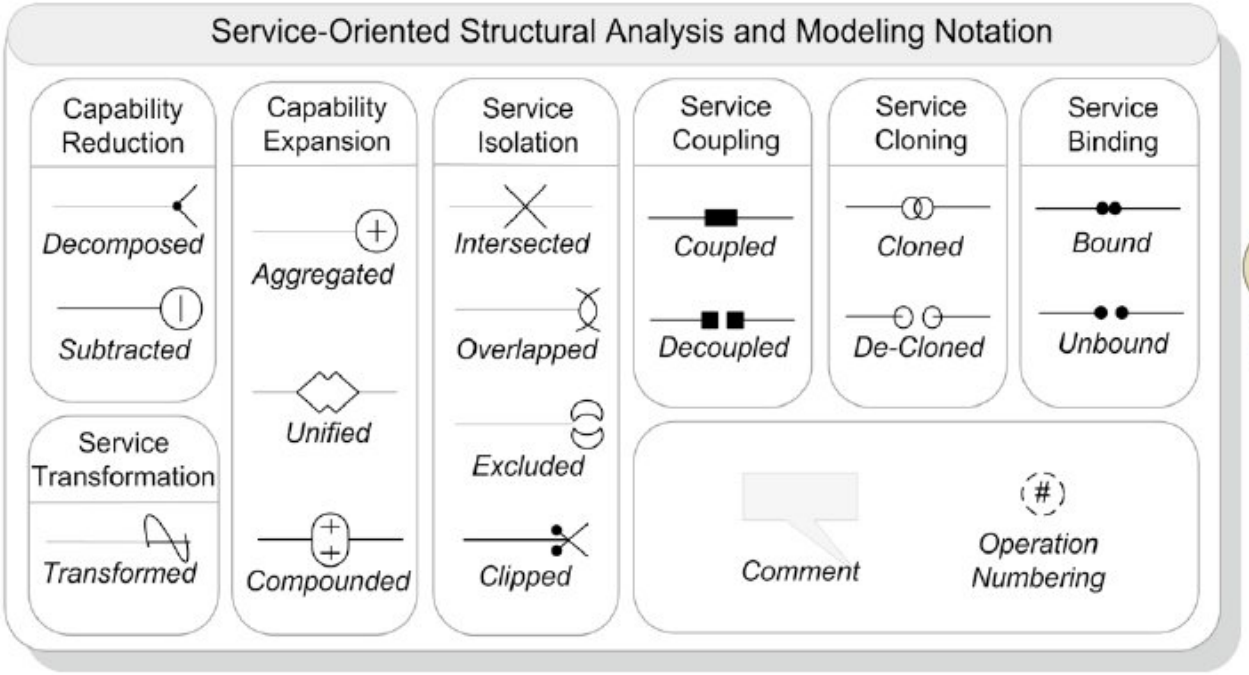
Service Oriented Modelling Framework



Service Oriented Analysis

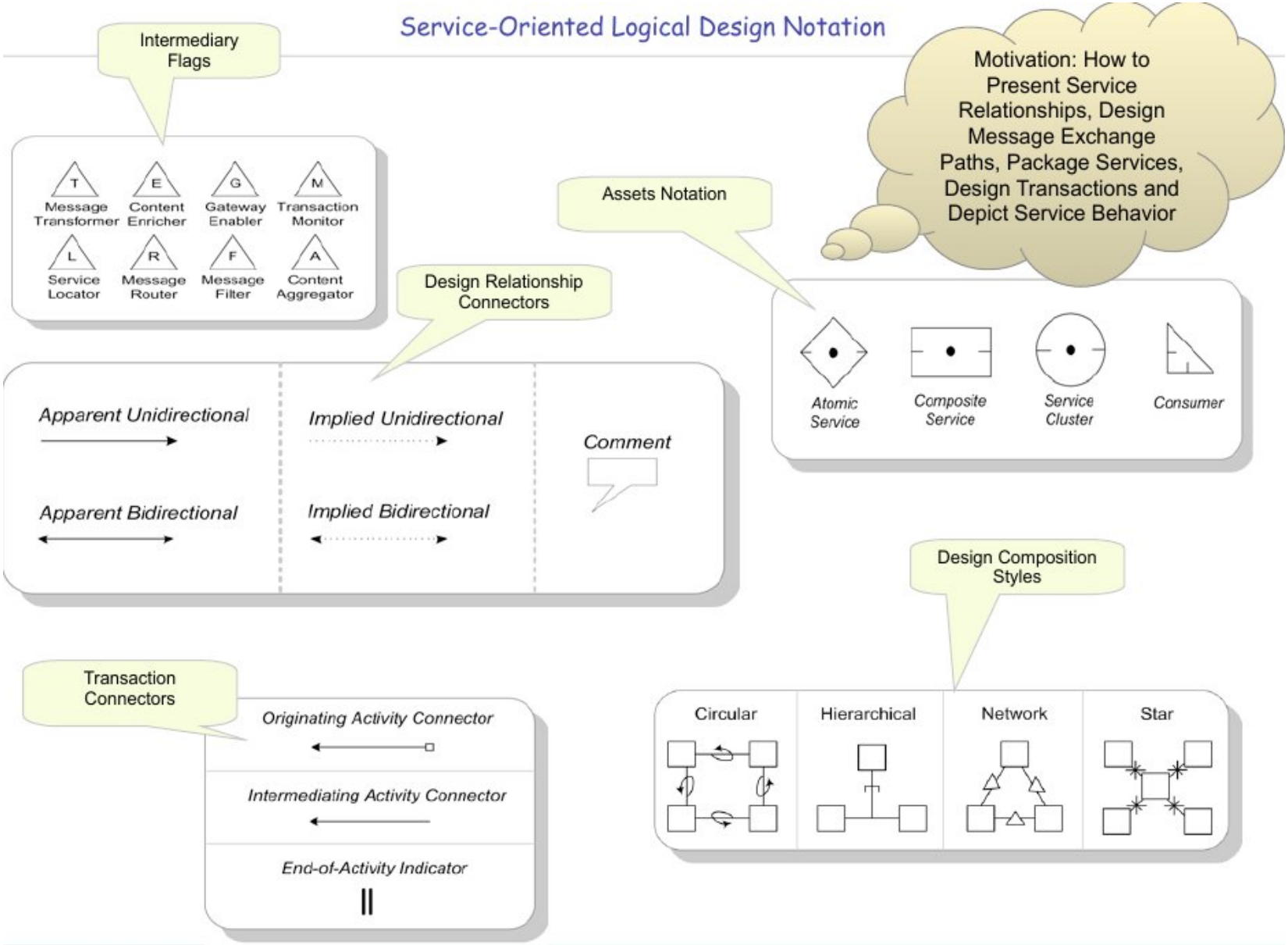






Motivation: How to Discover and Analyze services for Granularity, Reusability, Interoperability Success, etc…

Service Oriented Logical Design



Bisnis Proses Layanan SPBE

- Level 0: Penjabaran “Service-service”
- Level 1: Penjabaran proses yang ada untuk memberikan service-service tersebut, informasi yang dijelaskan hanya pada input/output proses tersebut
- Level 2: Penjabaran sub-proses dari proses tersebut beserta informasi/data yang digunakan pada sub-proses tersebut

Tingkatan Bisnis Proses

Bisnis Proses Organisasi
Contoh : Bispro BKN

Level: 0 semua aktifitas Di organsasi

Level 1: Layanan yang ada

Level 2: Proses

Bisnis Proses Layanan
Contoh: Layanan Kepegawaian

Level 0: Layanan yang ada

Level 1: Proses

Level 2: Sub-Proses

Information Modelling

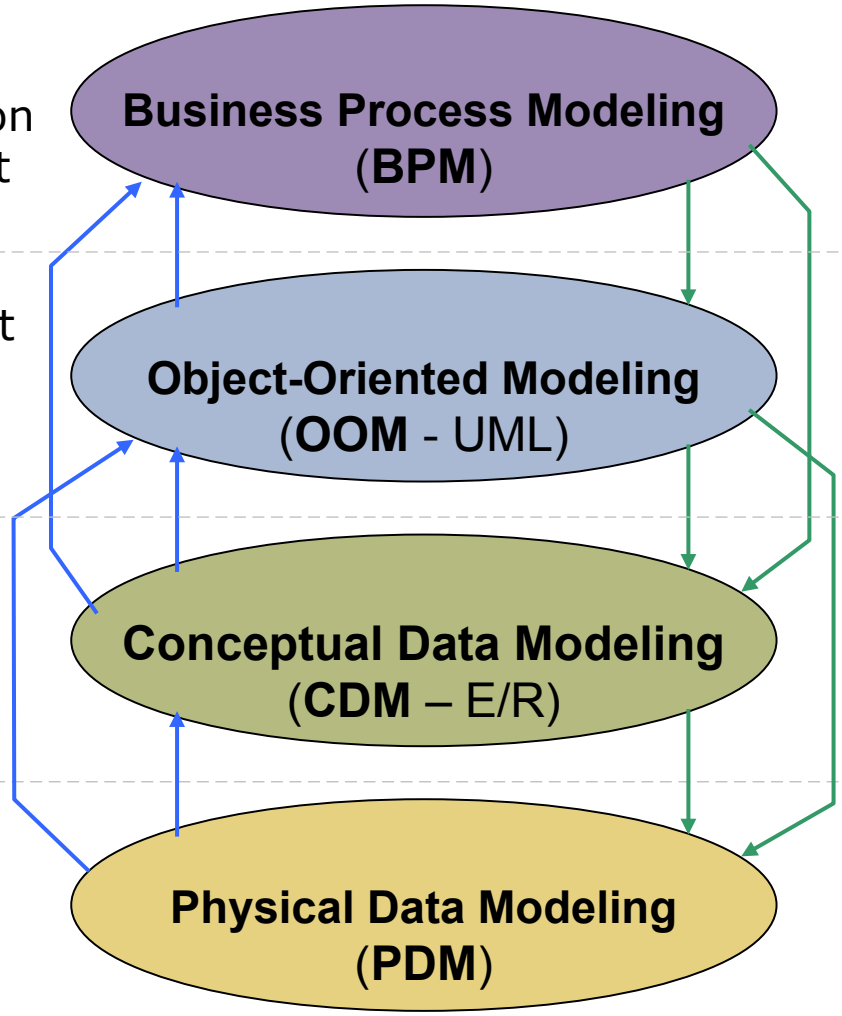
Data Modelling

Deskripsi

- **No Proses** : Level-Aktor_Pelaksana-No_Proses
- **Nama Proses:** Dimulai kata kerja transitif (yang membutuhkan obyek)
- **Tujuan**:
- **Ruang Lingkup**:
- **Input**:
- **Output**
- **CATATAN**: bila pada deskripsi ada DAN maka merupakan proses terpisah

World of Modeling ...

- End-user
- IT Partner/Liaison
- Business Analyst



- Improve process efficiency
- Define/document Bus. Processes
  - create correct and complete application **requirements**

- System Architect
- System Analyst
- App Developer

- Support for all **UML** diagrams
  - Analyze requirements
  - Design application
- Reverse/forward engineer code

- Bus. Analyst
- Data Architect
- Data Analyst

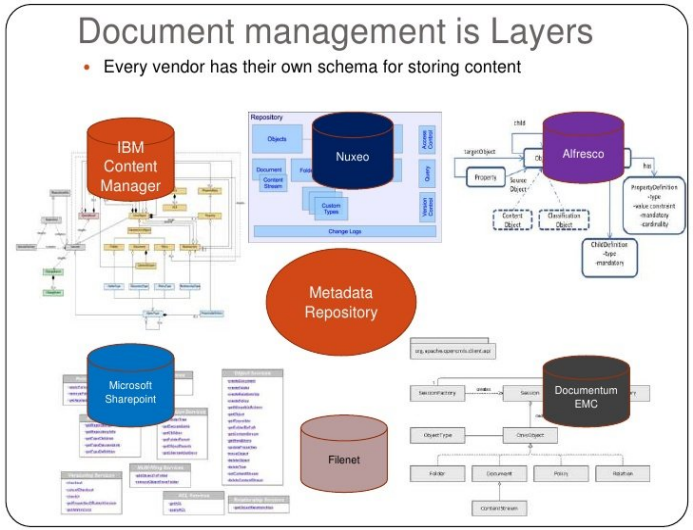
- Identify all data & relationships
  - **E/R** (Entity/Rel'ship) diagrams
  - DB independent view
- Business Rules?

- DBA
- DB Developer
- DB Architect

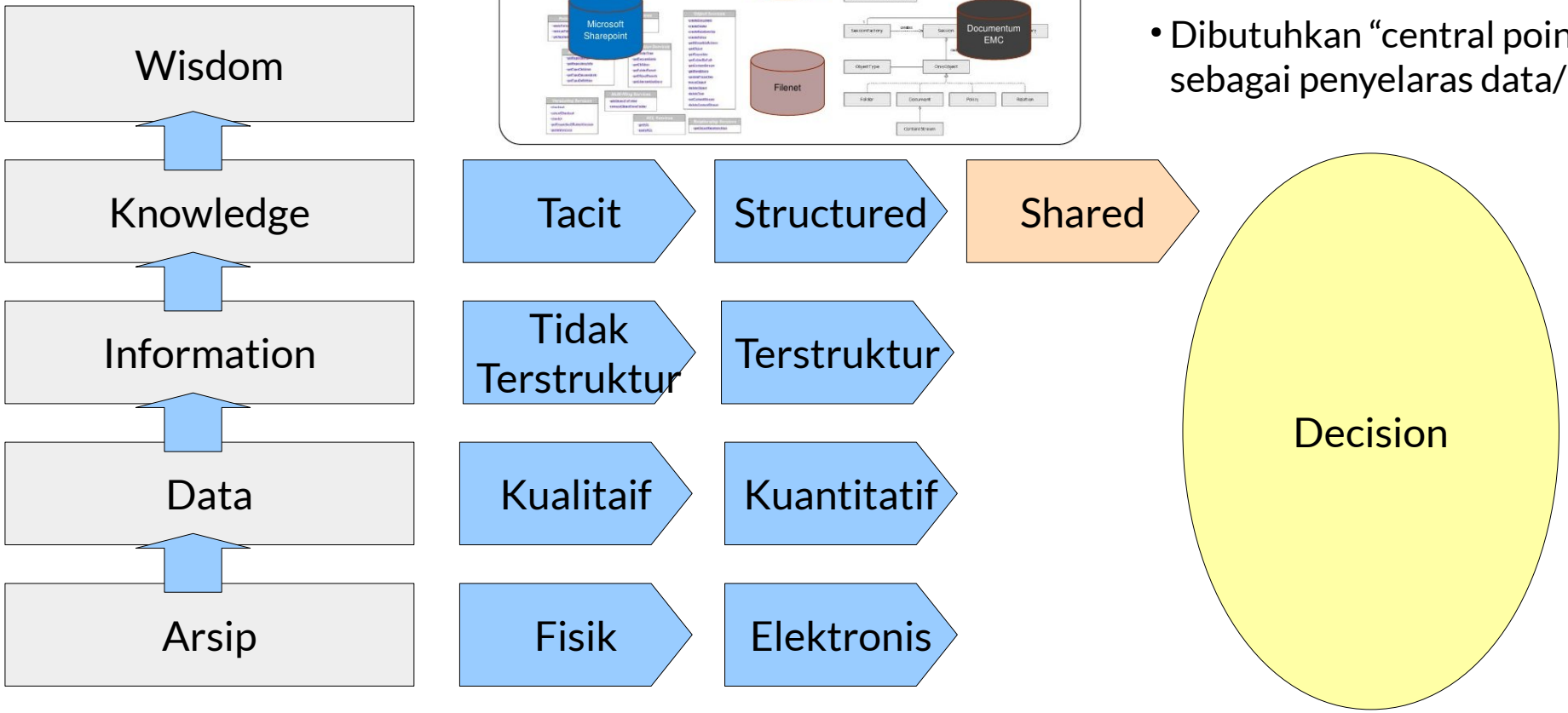
- DB-specific model
- **Reverse engineer** existing DB
- **Create/Update** DB from model
- Data Warehouse Modeling

QDesigner synchronizes models from all levels in a single tool

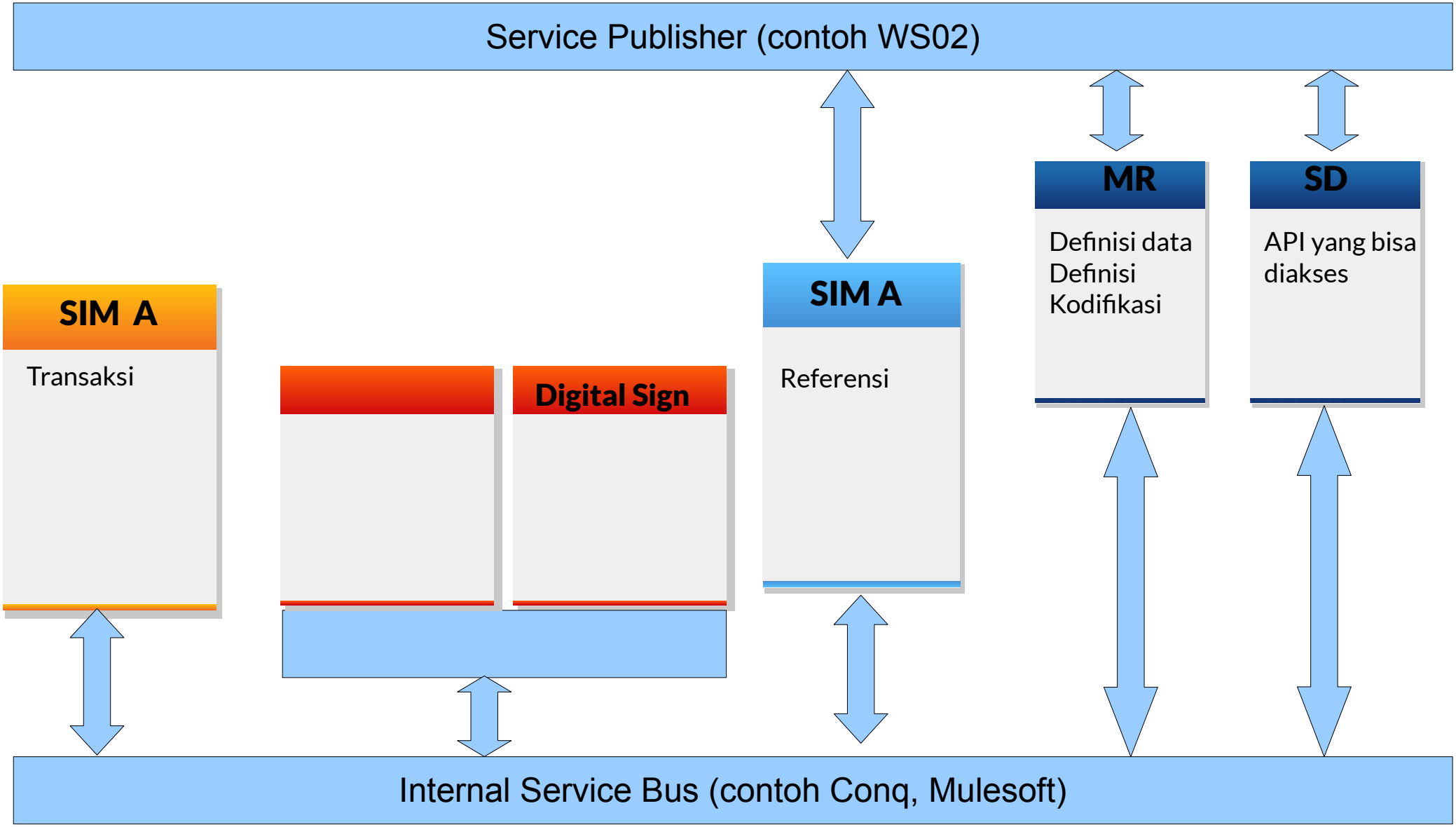
RI 4.0 berbasiskan Data - Informasi

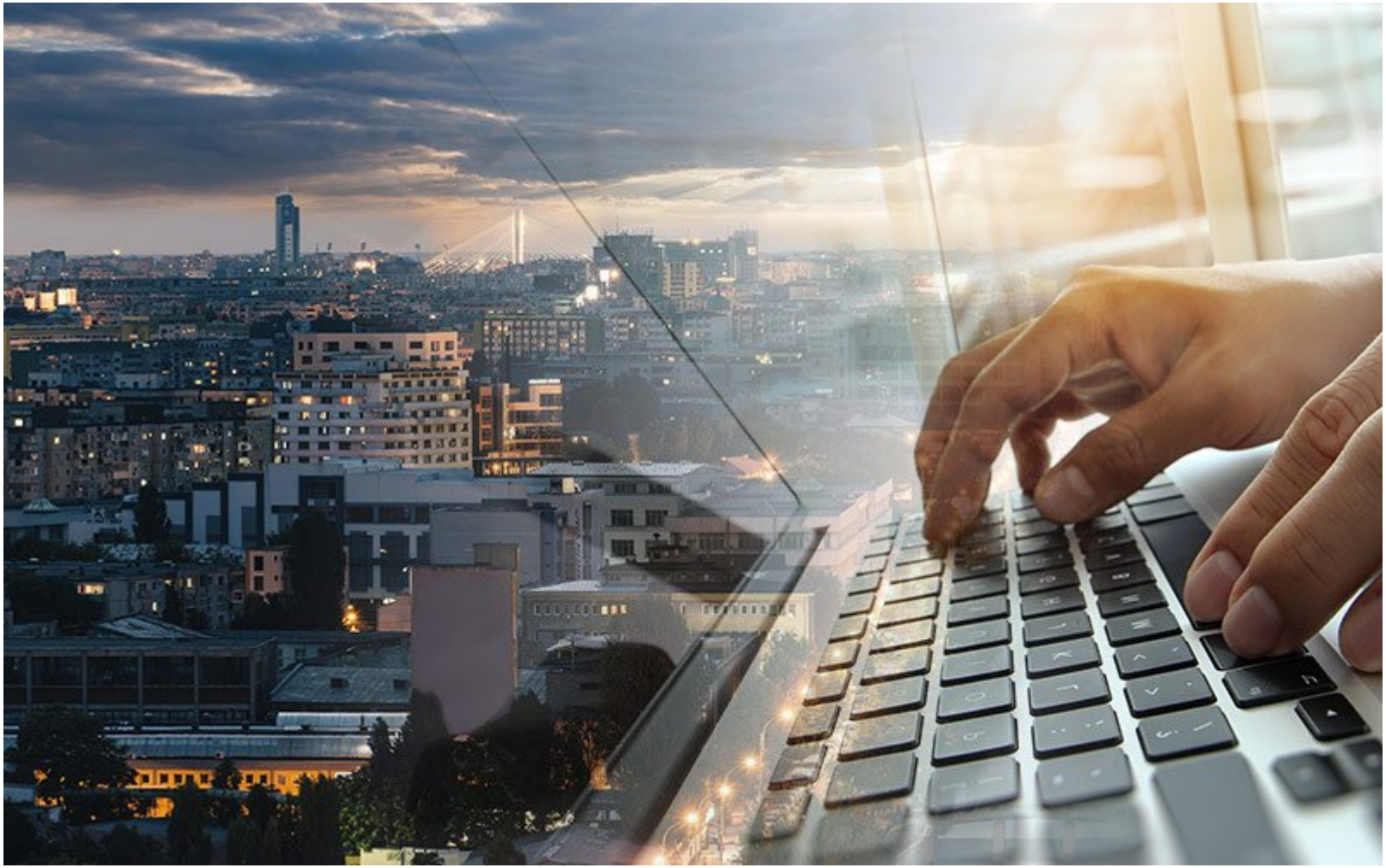


- Interoperabilitas di sistem yang heterogen
- Beragam organisasi dapat bertukar data/informasi
- Beragam teknologi dapat bertukar data/informasi
- Dibutuhkan "central point" sebagai penyelaras data/informasi



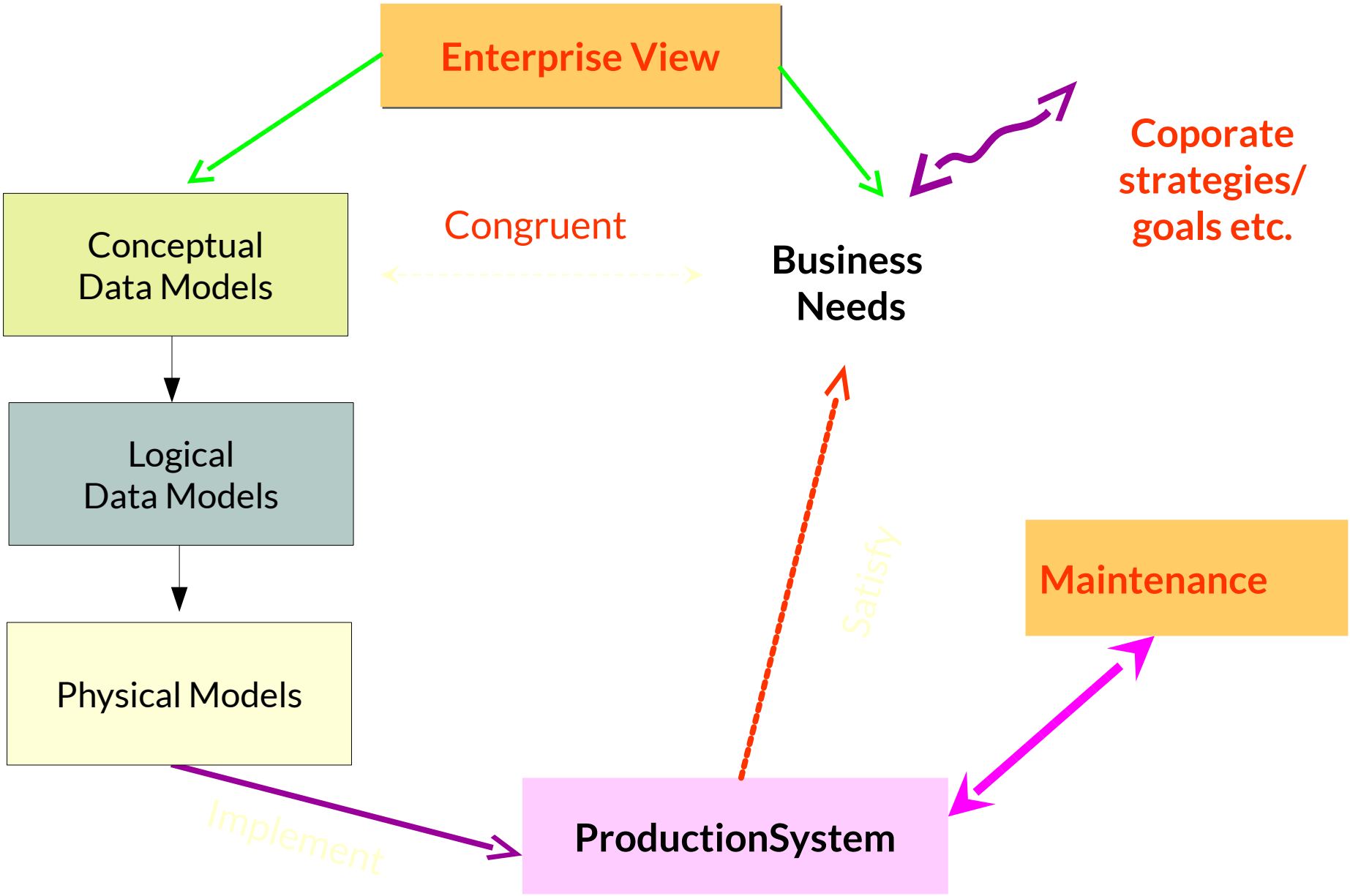
Rancang bangun





Persiapan Permodelan Data

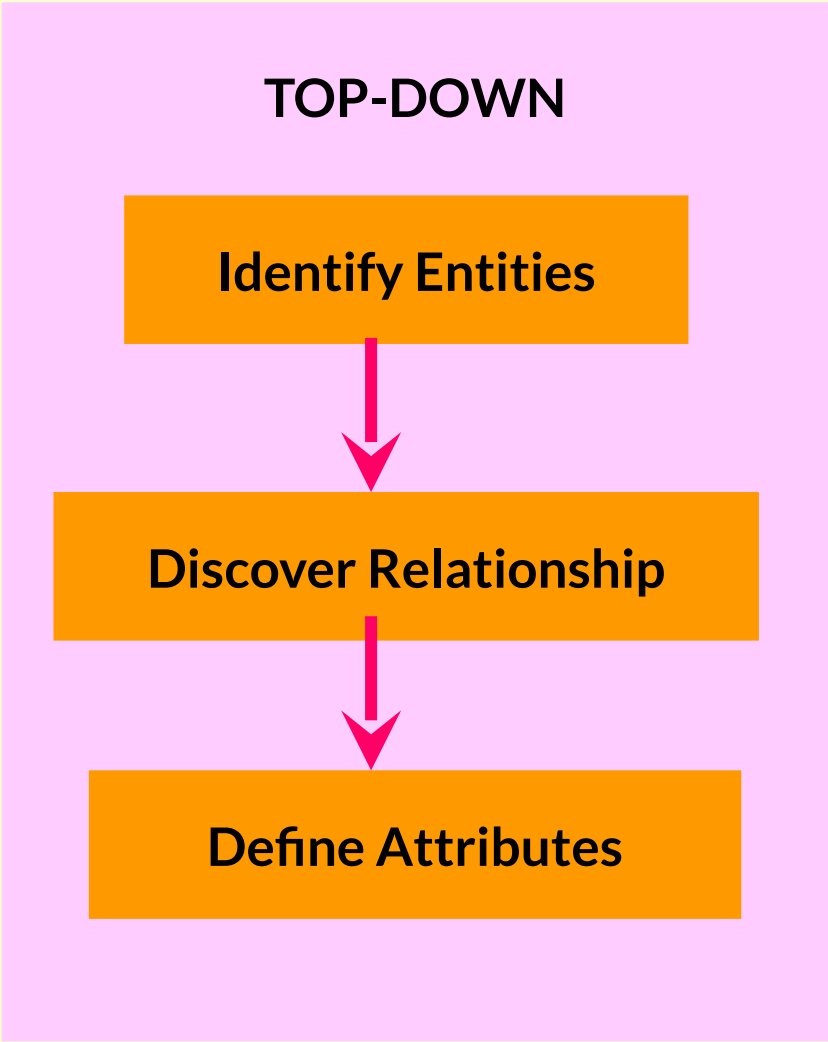
MODELING ENTERPRISE DATA

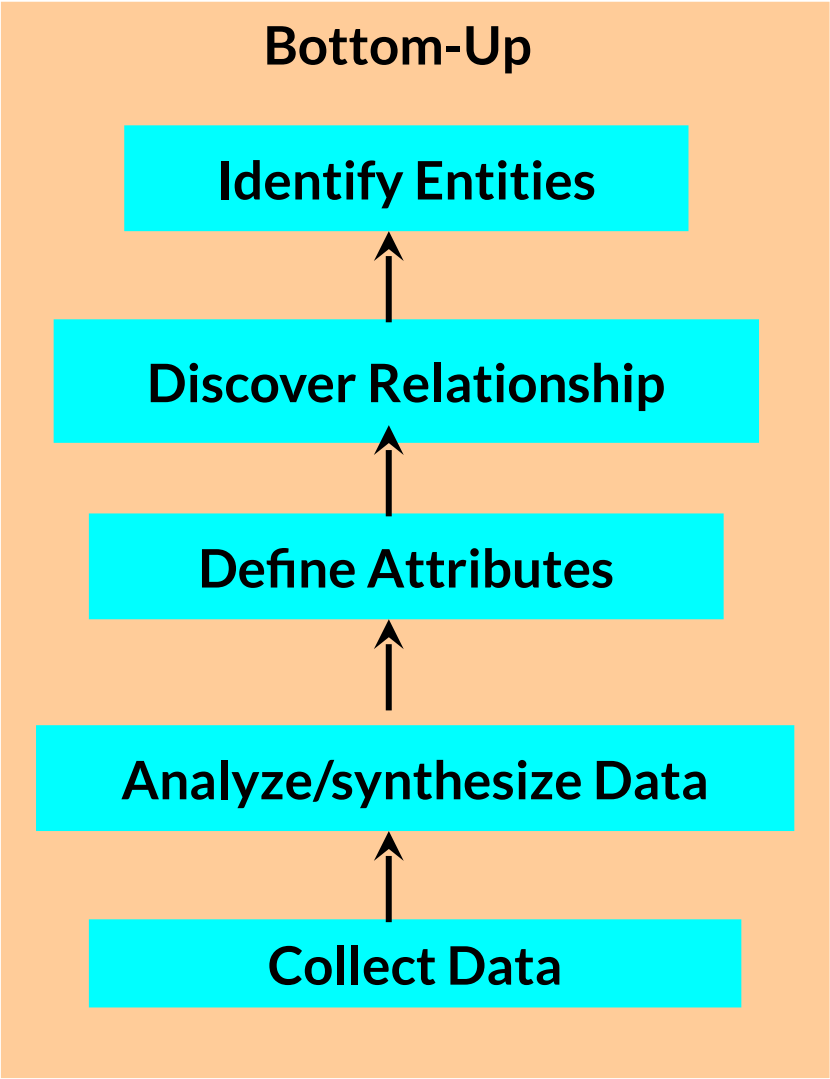


Approaches to Data Modeling

- Two approaches to data modeling:
  1. Top-down
  2. Bottom-up

Top-Down vs. Bottom-Up Approach





Survei Unit Kerja dan Tata Kelola

- Nama Unit Kerja
- Nama Jabatan Kepala Unit dan Person
- Jumlah SDM pada Unit Kerja tersebut
- Jumlah SDM bidang TIK
- SDM Pengelola data (memasukkan data, membuat laporan, memberikan data ke pihak lain)
- Jumlah Server yang dikelola: [on premise, atau collo]
- Jaringan sendiri yang dimiliki
- Tahapan apa saja yang dilakukan [Perencanaan, ….]

Survei Requirement

- Kebutuhan/harapan dari
  - SIM yang ada utk bekerja sehari-hari
  - Dashboard pada situation room bila ada (apa yang bisa dilihat pimpinan)
  - Hal-hal yang disukai pada SIM atau pengolahan data yang ada → preference
- Ketidakpuasan dari SIM pengolahan data
  - Hal-hal yang tidak disukai bila terjadi pada SIM atau pengolahan data → unsatisfactory factor utk node utama LWBA
  - Kasus yang pernah terjadi pada pengolahan data dan SIM → memahami "gap" dan "incident"

Survei SIM

- Nama SIM dan subdomain bila online:
- Unit pengelola:
- Lokasi instalasi SIM:
- Pengembang SIM :[internal, pihak ke-3, pusat]
- Fungsi dan Jenis SIM:
- Pengguna: [Internal unit, Internal Pemkab, Publik]
- Data-data yang ada pada SIM (bila perlu skema dan DATAnya)
- Data pihak eksternal yang dibutuhkan oleh SIM ini
- SIM eksternal yang dibutuhkan oleh pihak ini
- Dokumentasi yang ada untuk SIM ini:

Survei data elektronis/non elektronis

- Nama pengolahan data
- Unit kerja yang melakukan
- Format berkas: [wordprocess, spreadsheet, database], apa bisa memporelah
- Jenis informasi :[rahasia, terbatas, terbuka]
- Frekuensi update: berapa sering diupdate/dibuat
- Unit yang meminta data ini
- Data dari unit lain yang dibutuhkan untuk mengisi pengolahan data ini

RACI

- **Responsible**: adalah orang yang melakukan suatu kegiatan atau melakukan pekerjaan
- **Accountable:** adalah orang yang akhirnya bertanggung jawab dan memiliki otoritas untuk memutuskan suatu perkara
- **Consulted**: adalah orang yang diperlukan umpan balik atau sarannya dan berkontribusi akan kegiatan tersebut
- **Informed:** adalah orang yang perlu tahu hasil dari suatu keputusan atau tindakan

RACI Chart

Activities \ Functions	CEO	CFO	Business Executive	CIO	Business Process Owner	Head Operations	Chief Architect	Head Development	Head IT Administration	PMO	Compliance, Audit, Risk and Security
Translate data storage and retention requirements into procedures.				A	I	C	R				C
Define, maintain and implement procedures to manage the media library.				A		R	C	C	I		C
Define, maintain and implement procedures for secure disposal of media and equipment.				A	C	R			I		C
Back up data according to scheme.				A		R					
Define, maintain and implement procedures for data restoration.				A	C	R	C	C			I

A **RACI** chart identifies who is **R**esponsible, **A**ccountable, **C**onsulted and/or **I**nformed.

Survei Laporan

- Nama Laporan
- Acuan Aturan: UU/PP/PM atau peraturan lainnya yang mewajibkan aturan ini disusun, baik isi maupun format (contoh Asset Daerah, dengan KIP, Keuangan dsb)
- Frekuensi: (berapa bulan/minggu sekali)
- Unit pembuat laporan:
- Unit peminta laporan:
- Unit yang membutuhkan laporan sebagai informasi acuan

Inventarisasi Informasi

Informasi	Jenis (String, Numeris..)	Frek update	Dependency	Klasifikasi (Rahasia/ Terbatas/ Publik)	Aturan/ Konteks

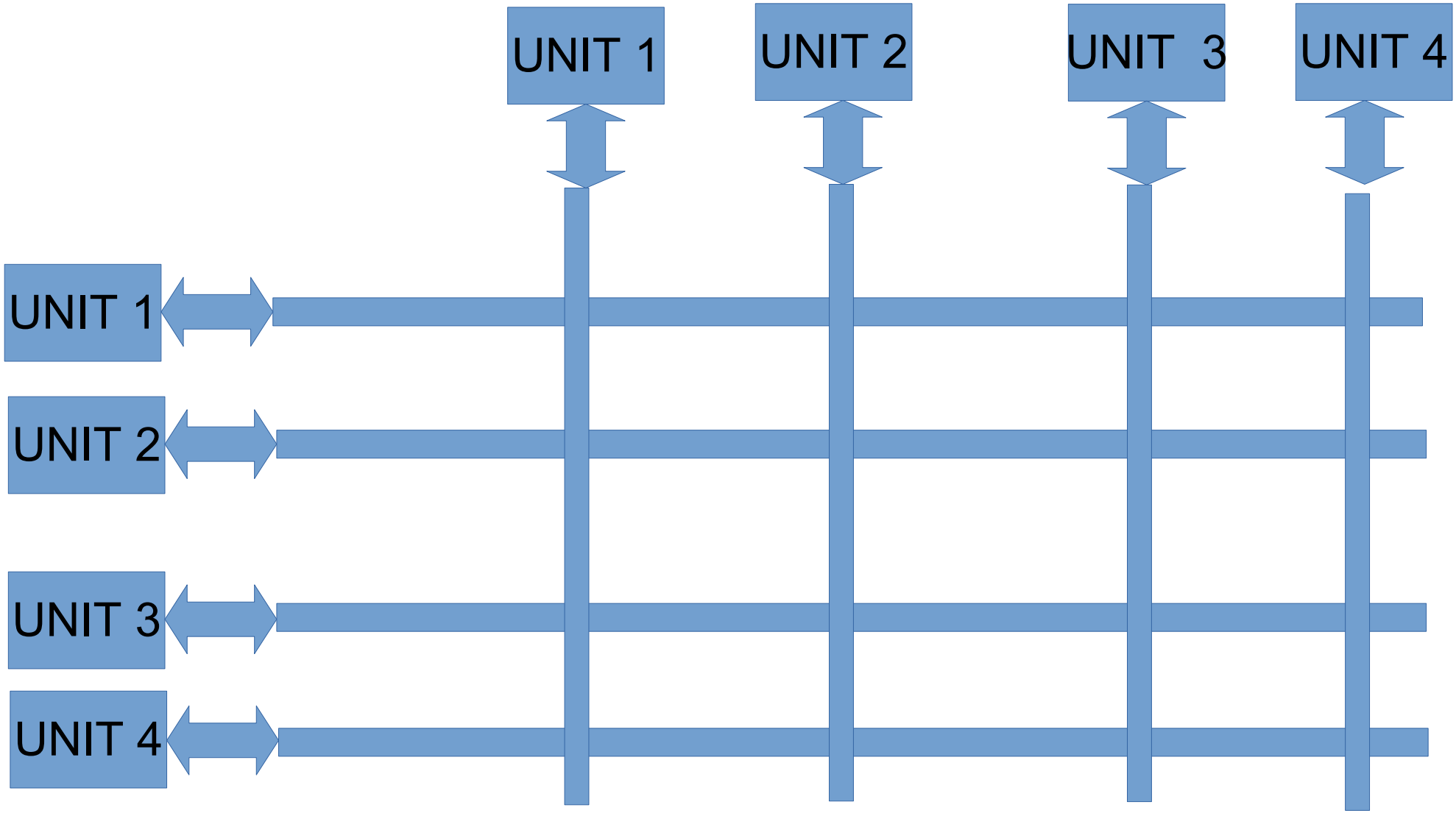
- Informasi: jenis informasi
- Jenis: jenis data penyimpan informasi dan besarnya
- Frek Update: berapa sering informasi di-update
- Kebergantungan pada informasi lain (mengupdate apa- diupdate apa)
- Klasifikasi keterbukaan : apakah bersifat rahasia/terbatas/publik
- Klasifikasi kekritisan: : sejauh mana mengganggu jalannya business proses: kritis/agak kritis/tidak kritis
- Aturan: aturan yang membahas pengaturan informasi ybs: wajib, atau rutin wajib, paling telat berapa lama

Inventarisasi Service

Service	Sumber data	Format	Frekuensi	Dependency	Klasifikasi (Rahasia/ Terbatas/ Publik)	Aturan/ Konteks

- Service : jenis layanan yang diberikan
- Sumeber data:organisasi pengelola layanan tersebut
- Format : format data yang diberikan oleh service tersebut
- Frek Update: berapa sering informasi tersebut diminta
- Kebergantungan pada service lain (mengupdate apa- diupdate apa)
- Klasifikasi keterbukaan: apakah bersifat rahasia/terbatas/publik
- Klasifikasi kekritisan: : sejauh mana mengganggu jalannya business proses: kritis/agak kritis/tidak kritis
- Aturan: aturan yang membahas pengaturan informasi ybs: wajib, atau rutin wajib, paling telat berapa lama

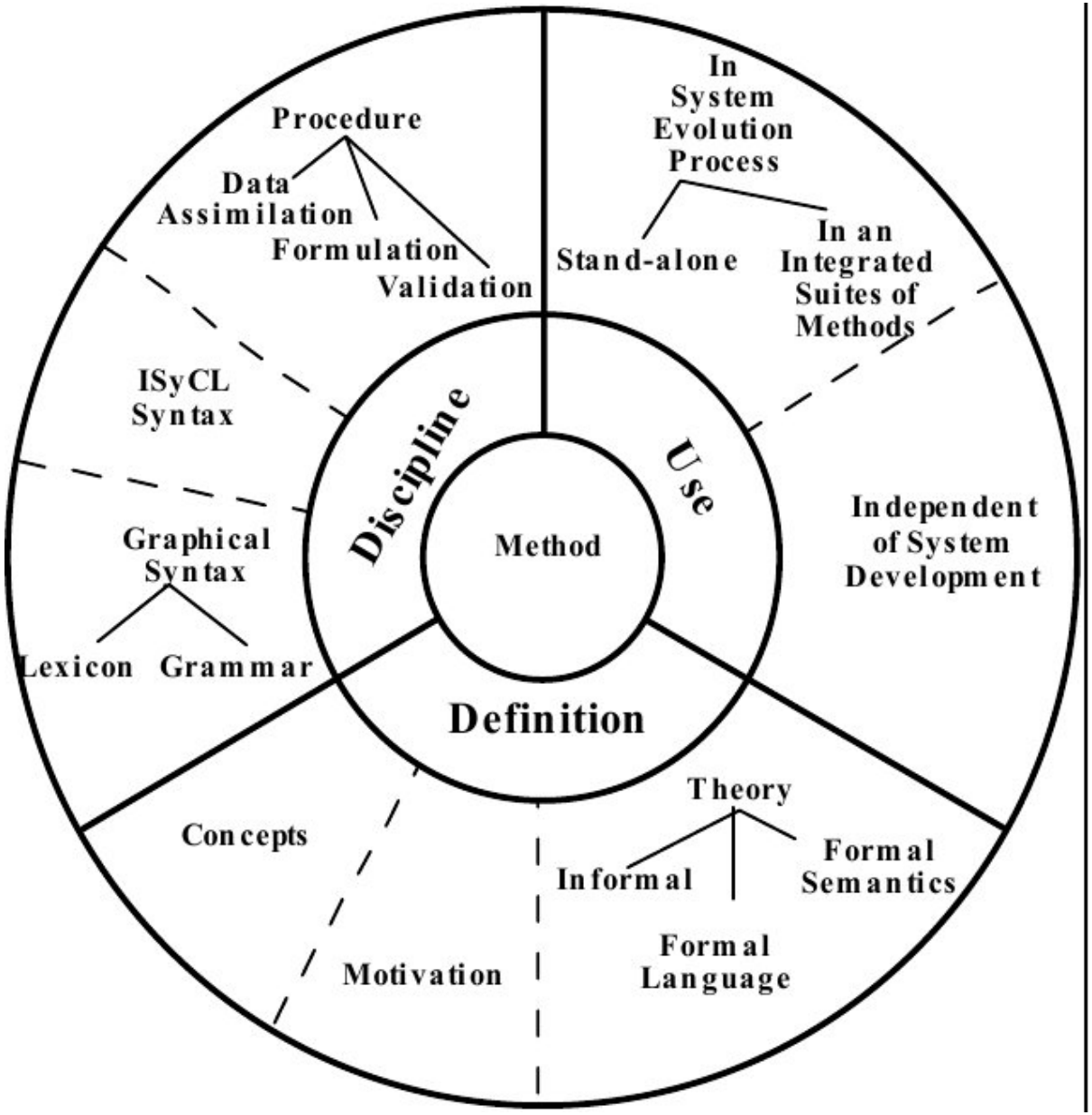
Matrix kebutuhan informasi



Survei Integrasi

- Model integrasi
  - Apakah sudah menerapkan integrasi secara federated melalui:
    - Integrasi service
    - Intergrasi tabel data
    - Integrasi tampilan
  - Bagaimana mekanisme sinkronisasi data
    - Secara rutin per waktu
    - Ketika ada perubahan di satu data maka dilakukan sinkronisasi
    - Sinkronisasi manual
- Komponen pendukung integrasi. Apakah telah diterapkan
  - Menggunakan Service Bus
- Permasalahan integrasi data
  - Kasus yang pernah terjadi pada pengolahan data dan SIM → memahami "gap" dan "incident"

Komponen Modelling



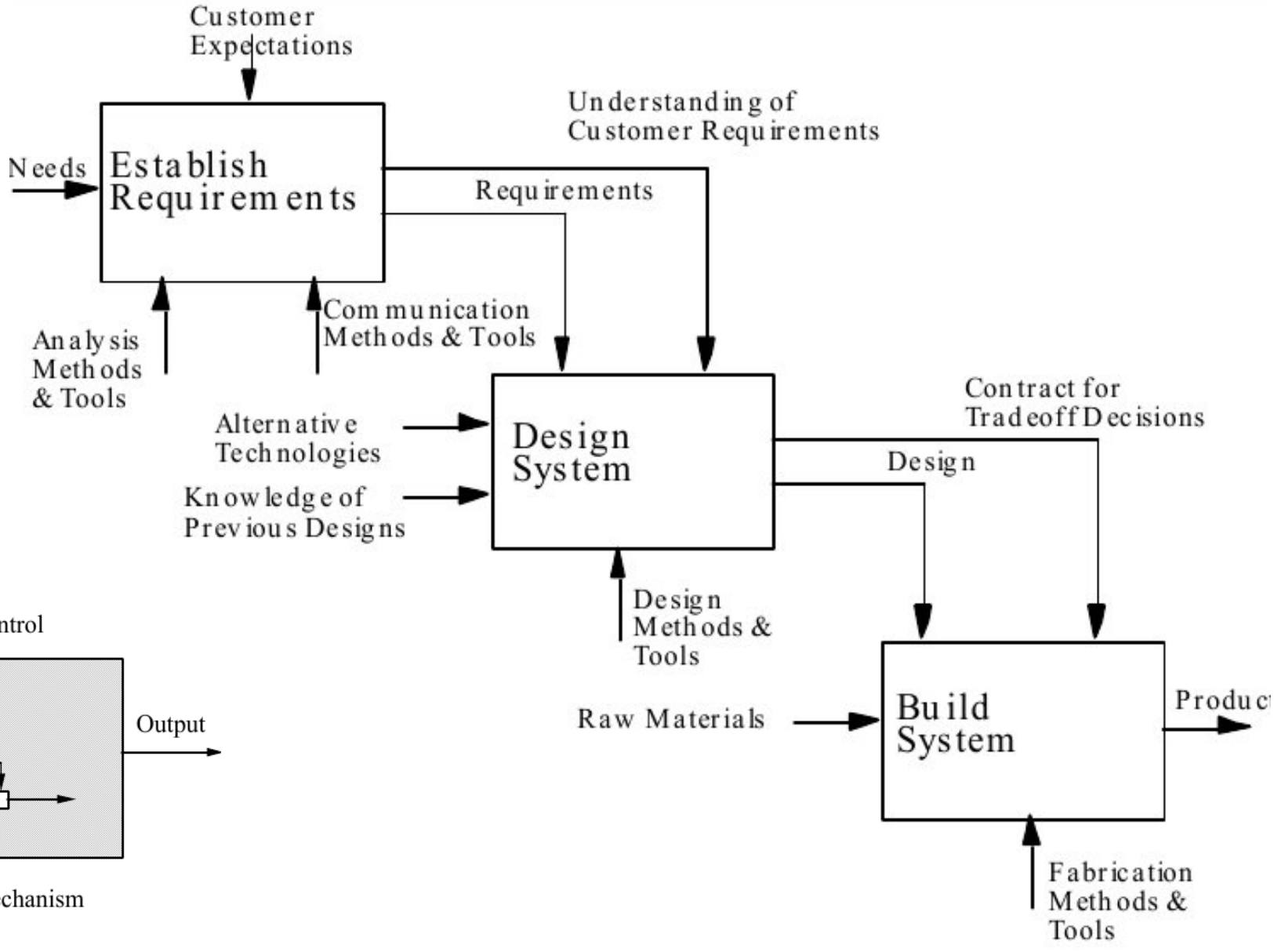
IDEF

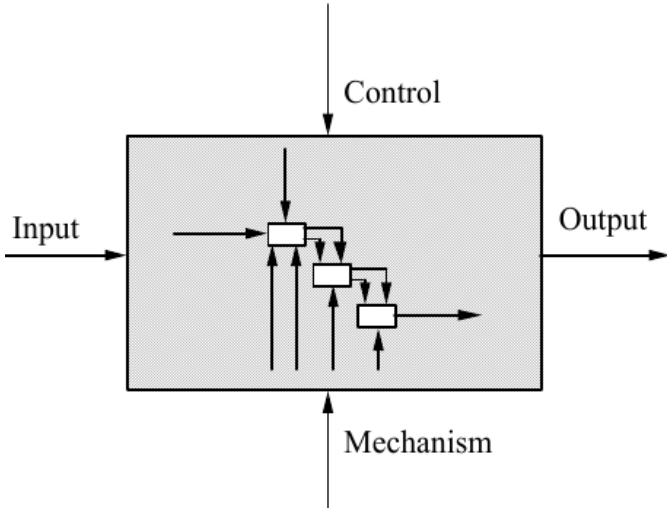
- Awalnya singkatan dari ICAM Definition, tapi di 1999 menjadi Integration DEFinition
- Merupakan satu kumpulan bahasa permodelan pada rekayasa perangkat lunak
- Dikembangkan oleh US Airfoce dan banyak digunakan di USA DoD dan merupakan public domain
- ISO/IEC/IEEE 31320-1:2012(en) → Information technology — Modeling Languages — Part 1: Syntax and Semantics for IDEF0
- ISO/IEC/IEEE 31320-2:2012(en) → Information technology — Modeling Languages — Part 2: Syntax and Semantics for IDEF1X97 (IDEFobject)
- Hingga tahun 1995 hanya IDEF0, IDEF1x, IDEF2, IDEF4 telah dikembangkan lengkap. Beberapa IDE sudah mulai dirancang. Pada tahun 1995 IDEF9, IDEF6, IDEF8 dan IDF 14 sudah dikegkangkan
- Metoda IDEF7, IDEF10, IDEF11, IDEF12, IDEF14 belum dikembuna

Jenis-jenis IDEF

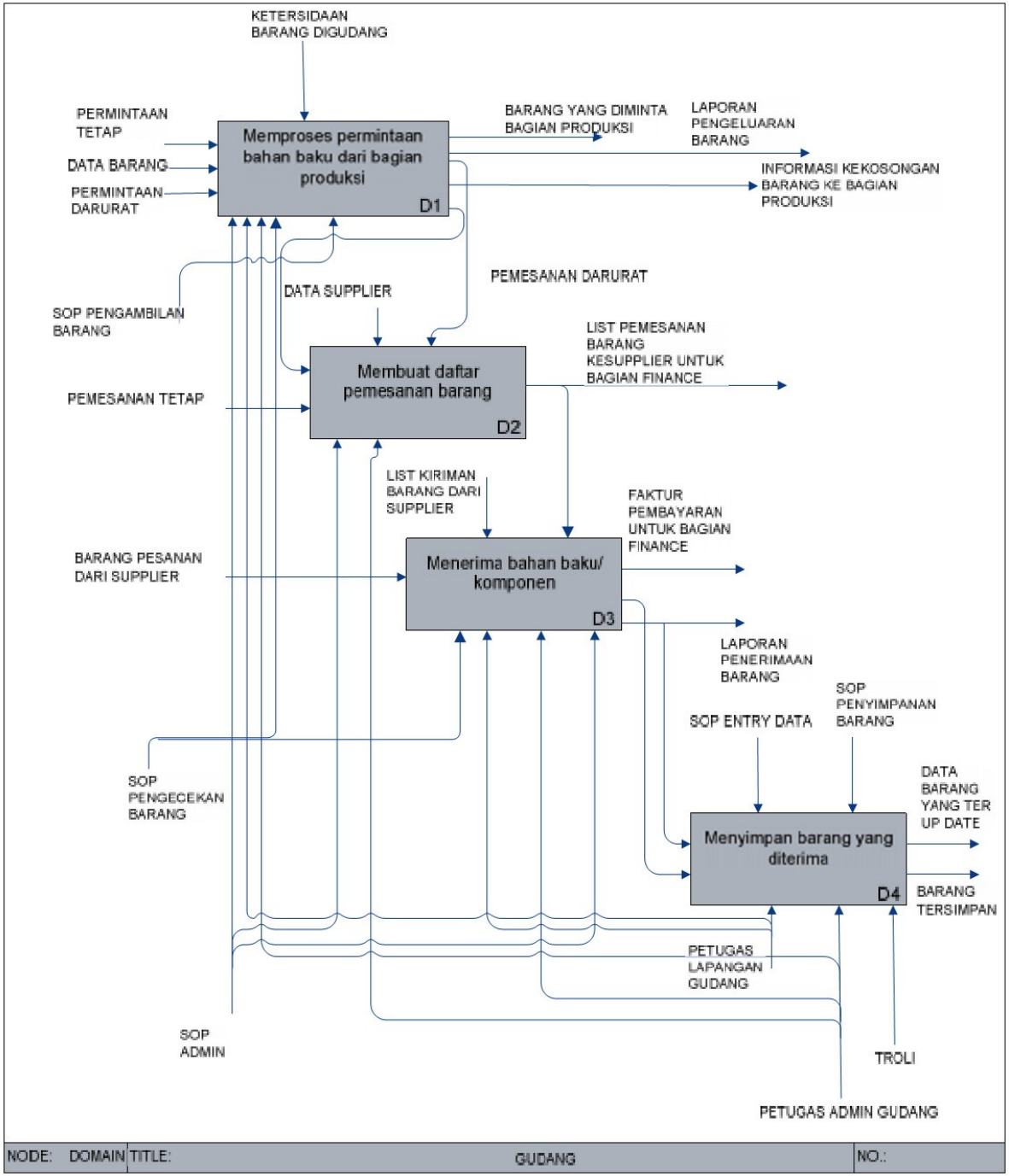
- IDEF0 : Function modeling[3]
- IDEF1 : Information modeling [4]
- IDEF1X : Data modeling [5]
- IDEF2 : Simulation model design
- IDEF3 : Process description capture [6]
- IDEF4 : Object-oriented design [7]
- IDEF5 : Ontology description capture [8]
- IDEF6 : Design rationale capture [9]
- IDEF7 : Information system auditing
- IDEF8 : User interface modeling
- IDEF9 : Business constraint discovery
- IDEF10 : Implementation architecture modeling
- IDEF11 : Information artifact modeling
- IDEF12 : Organization modeling
- IDEF13 : Three schema mapping design
- IDEF14 : Network design

IDEF0 pembuatan sistem



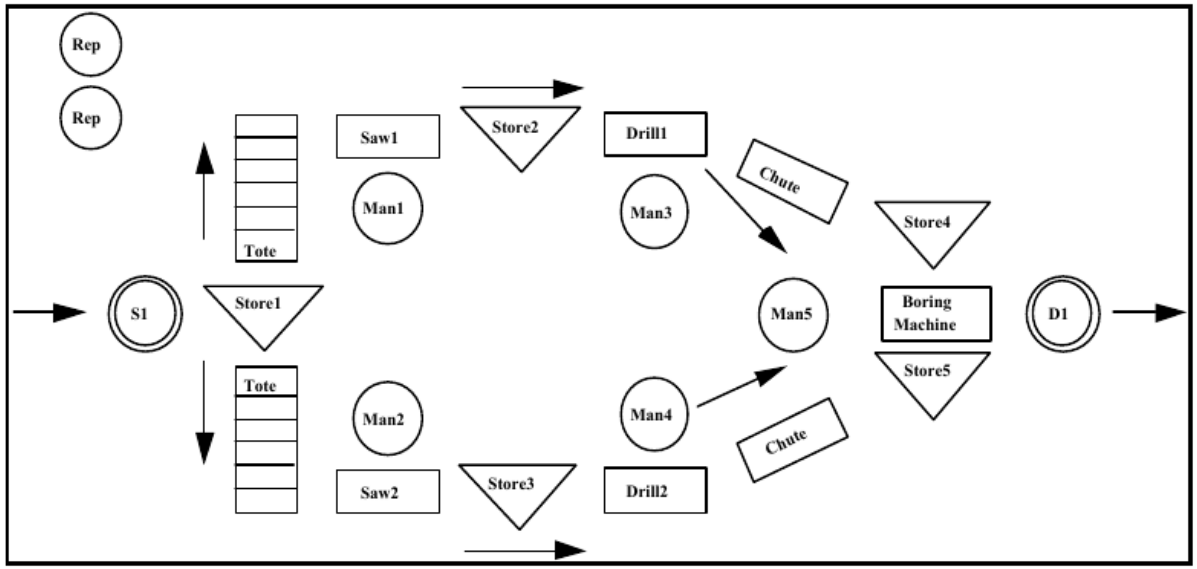


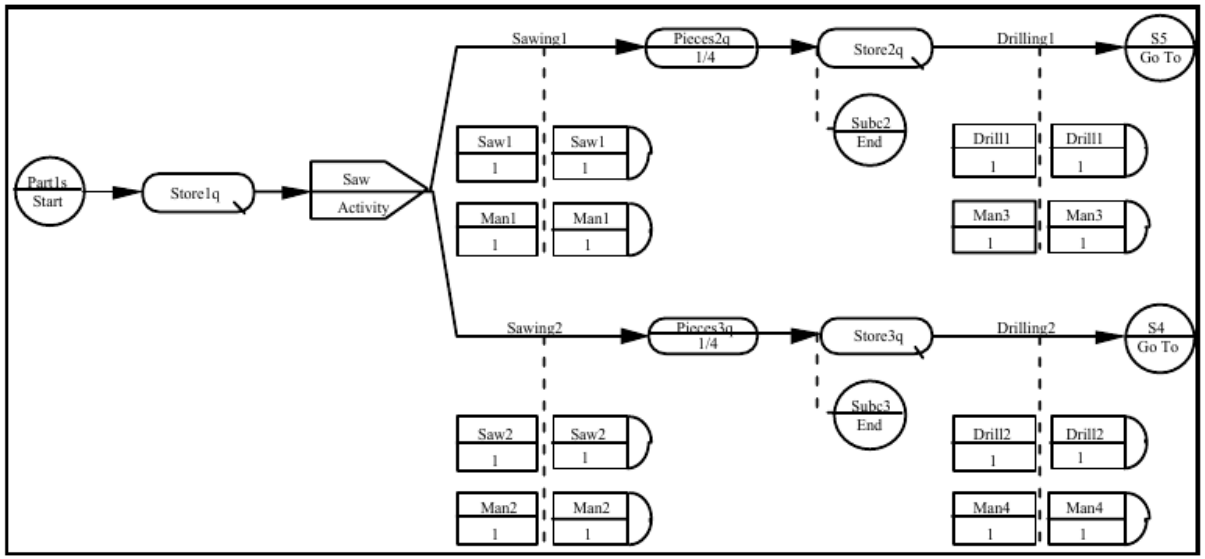
IDEF0


KETERSIDAAN BARANG DIGUDANG
PERMINTAAN TETAP
DATA BARANG
PERMINTAAN DARURAT
Memproses permintaan bahan baku dari bagian produksi
D1
BARANG YANG DIMINTA BAGIAN PRODUKSI
LAPORAN PENGELUARAN BARANG
INFORMASI KEKOSONGAN BARANG KE BAGIAN PRODUKSI
SOP PENGAMBILAN BARANG
DATA SUPPLIER
PEMESANAN DARURAT
Membuat daftar pemesanan barang
D2
LIST PEMESANAN BARANG KESUPPLIER UNTUK BAGIAN FINANCE
PEMESANAN TETAP
LIST KIRIMAN BARANG DARI SUPPLIER
FAKTUR PEMBAYARAN UNTUK BAGIAN FINANCE
BARANG PESANAN DARI SUPPLIER
Menerima bahan baku/ komponen
D3
LAPORAN PENERIMAAN BARANG
SOP PENYIMPANAN BARANG
SOP ENTRY DATA
SOP PENGECEKAN BARANG
DATA BARANG YANG TER UP DATE
Menyimpan barang yang diterima
D4
BARANG TERSIMPAN
PETUGAS LAPANGAN GUDANG
SOP ADMIN
TROLI
PETUGAS ADMIN GUDANG
NODE:
DOMAIN
TITLE:
GUDANG
NO.:

IDEF2 – System Dynamic Modelling

- Facility Submodel (used to specify the model of the agents of the system and environment).
- Entity Flow Submodel (used to specify the model of the transformations that an entity undergoes).
- Resource Disposition Submodel (specifies the logic of agent assignment to entity transform demands).
- System Control Submodel (specifies the effect of transformation independent or system external events on the modeled system).

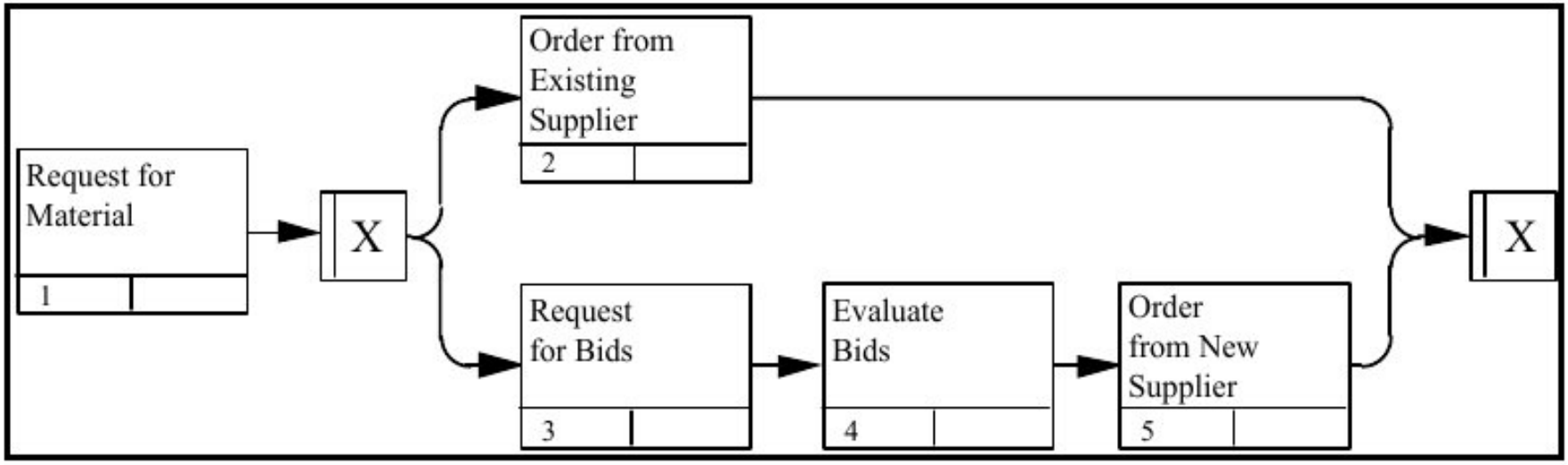




IDEF3 – Process Description

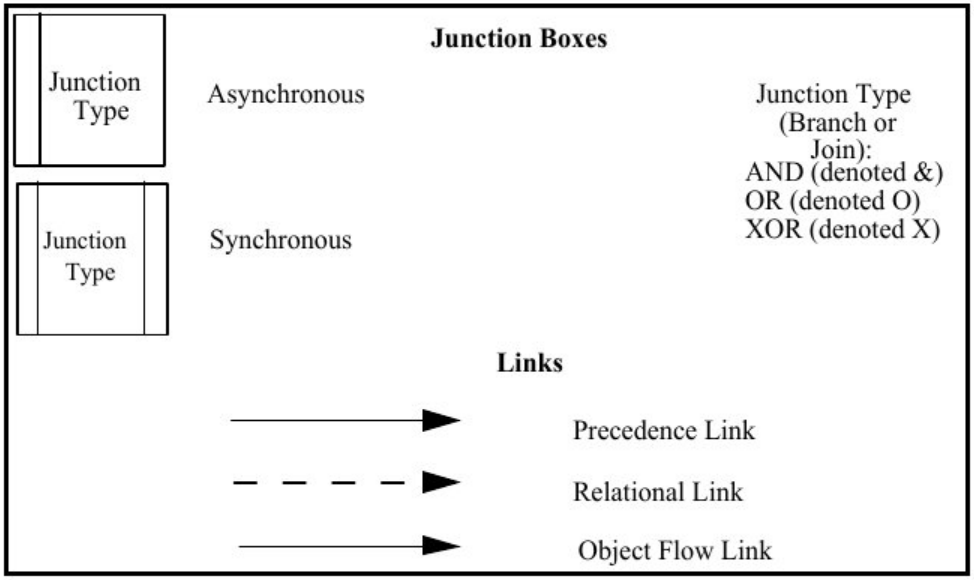
- To provide a systematic method for recording and analyzing the raw data that results from fact-finding interviews in a systems analysis
- project.
- To determine the impact of an organization's information resource on the major operating scenarios of an enterprise.
- To provide a mechanism for documenting the decision procedures affecting the states and life-cycle of critical shared data (particularly manufacturing, engineering, maintenance, and product definition data).
- To define data configuration management and change control policy definition.
- To support system design and design tradeoff analysis.
- To provide powerful mechanisms to support the generation of simulation models.
- To provide useful information for the creation of functional (IDEFØ) models.
- To facilitate process mapping for the design of software to achieve real-time control by providing a mechanism for clearly defining the facts, decision points, and job classifications.
- To provide an analyst with a method to clearly define the data needed to develop needs and requirements from a user viewpoint.
- To collect and express the views of domain experts required for the development of expert systems.

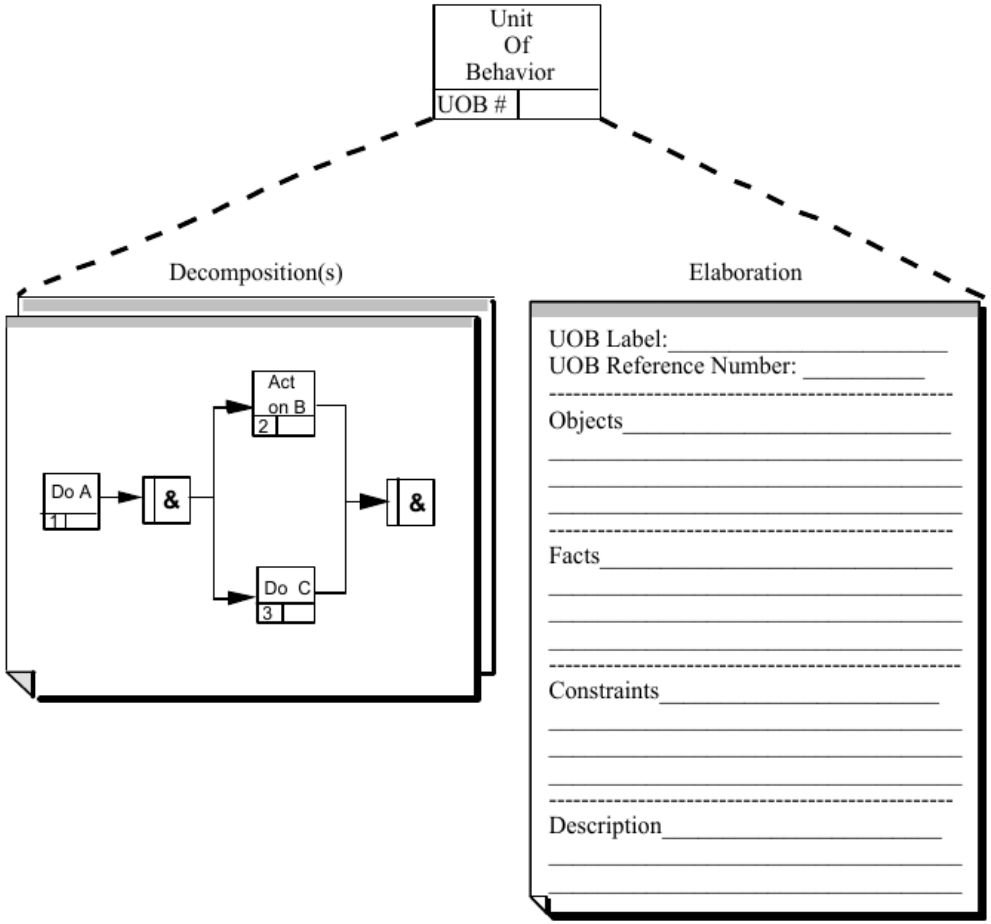
Diagram IDEF3



- Units of Behavior (UOBs)
- Junctions
- Links
- Referents
- Elaborations

Ditail Unit of Behaviour (UOB)





IDEF4 – Object Oriented Model

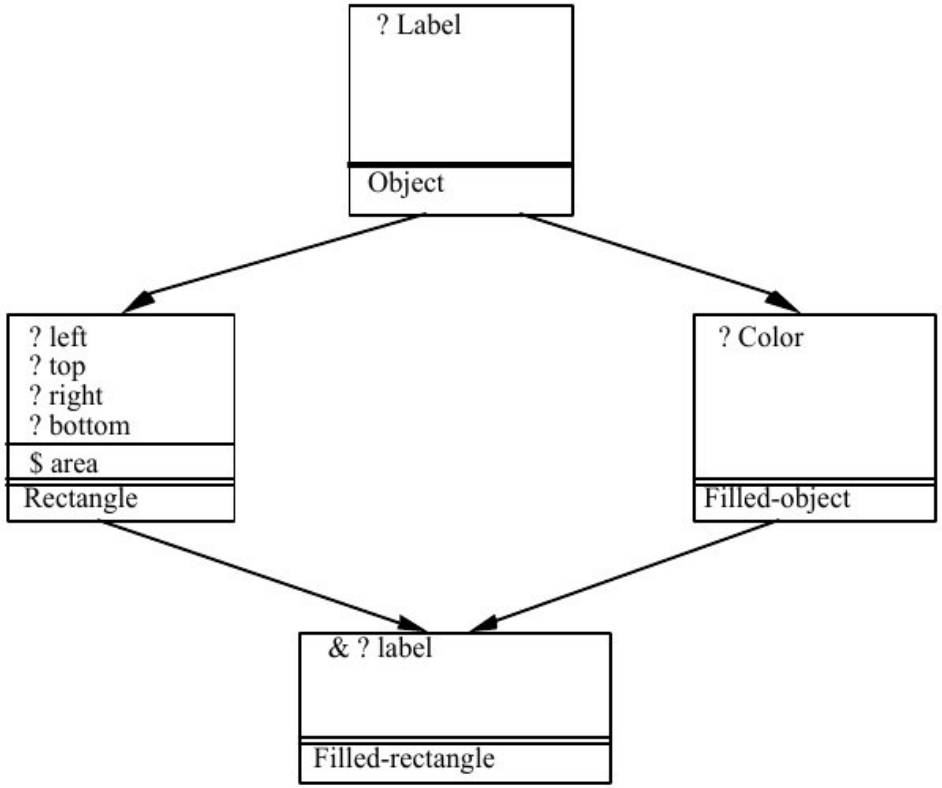


Figure 17. A Class Inheritance Diagram in IDEF4

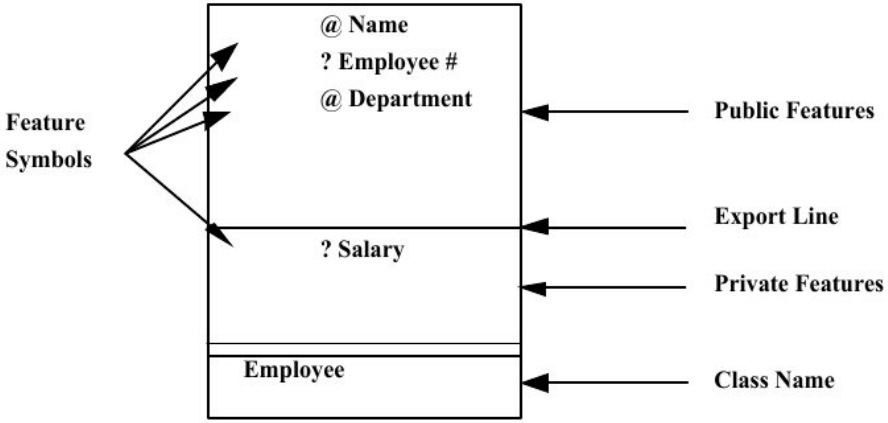


Figure 18. A Class Box in IDEF4

IDEF5 - Ontology

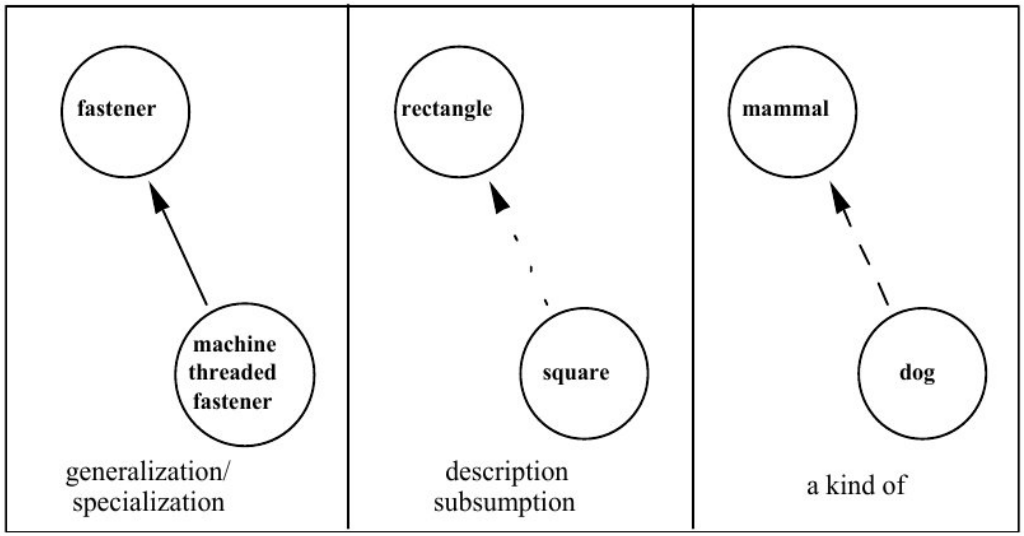
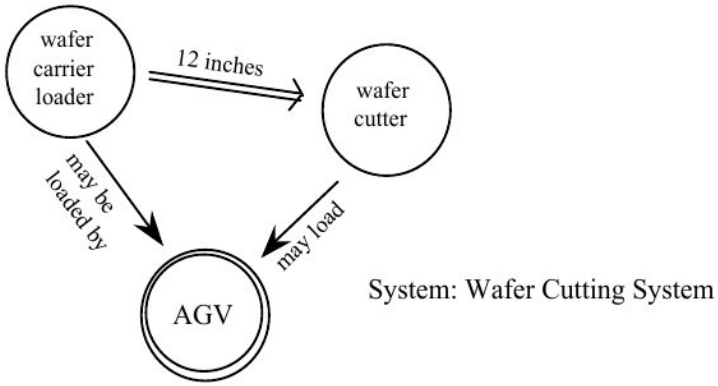
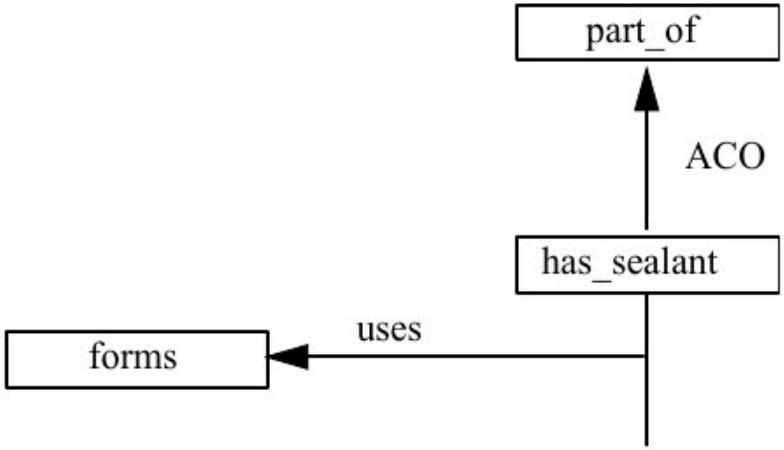


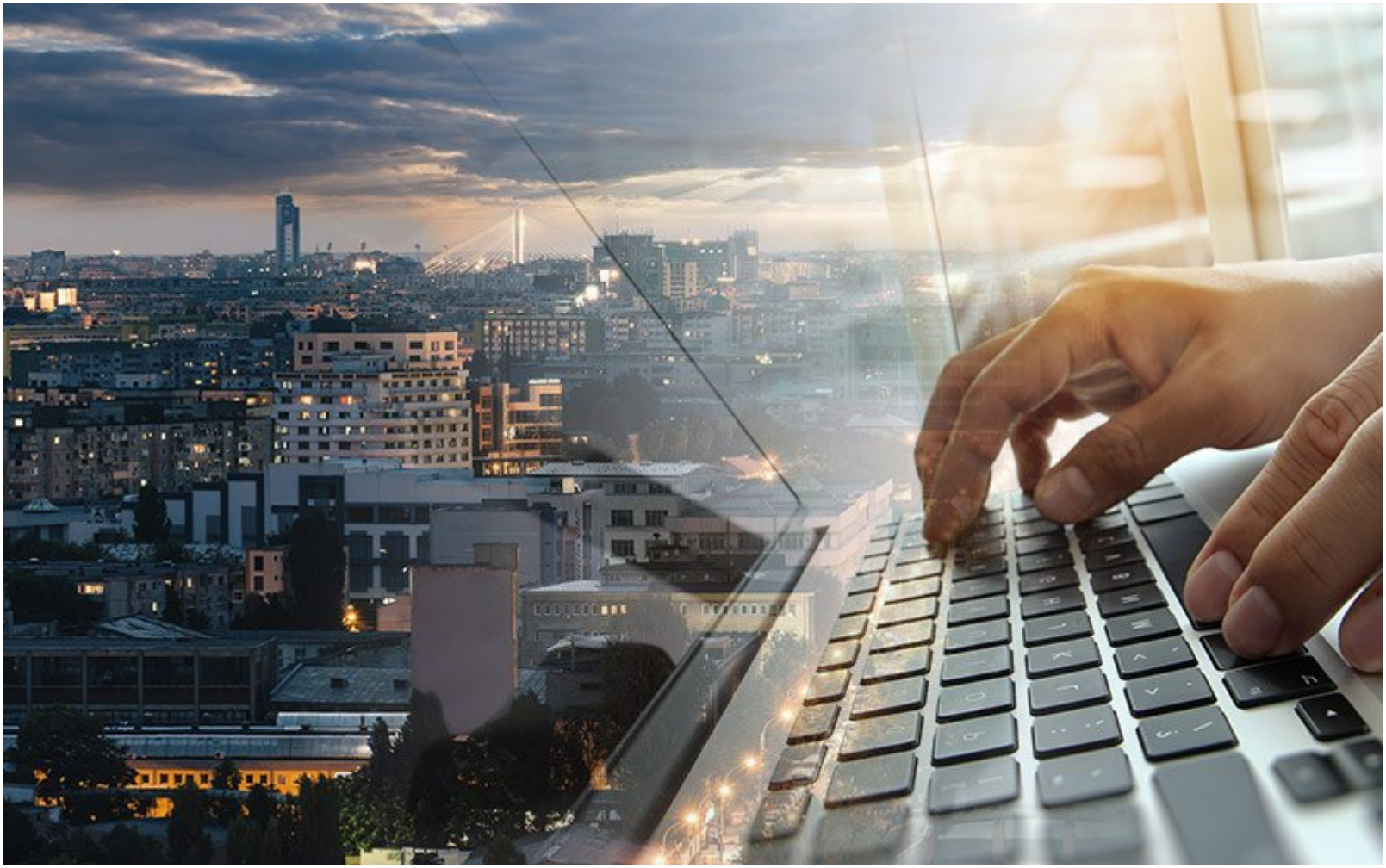
Figure 21. Is-a Link Types





Metoda yang disarankan

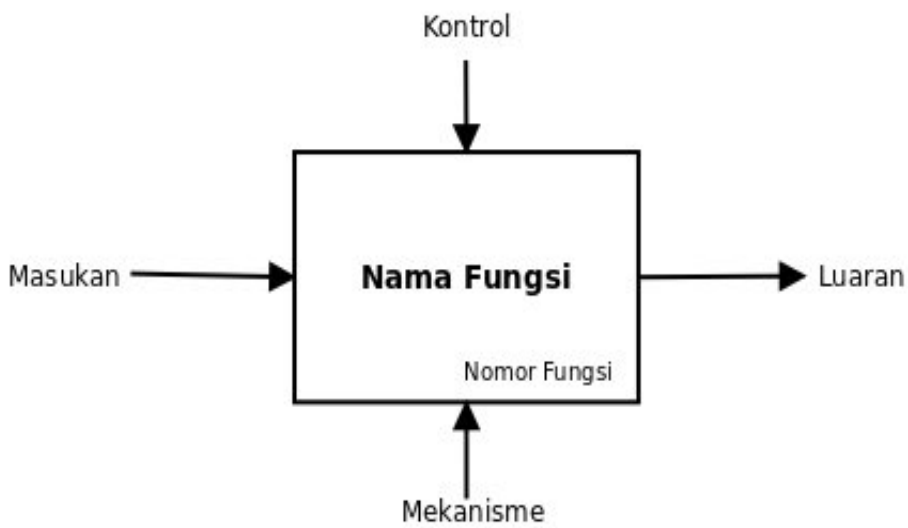
- Melakukan functional model → IDEF0
- Melakukan pemetaan fungsi → Use - Case
- Melakukan user model → Proto Persona
- Melakukan business process model → Bisa BPMN atau IDEF
- Melakukan information modeling → IDEF1
- Melakukan data modelling → IDEF1x
- Melakukan task modelling dan interaction modelling, tentukan pattern



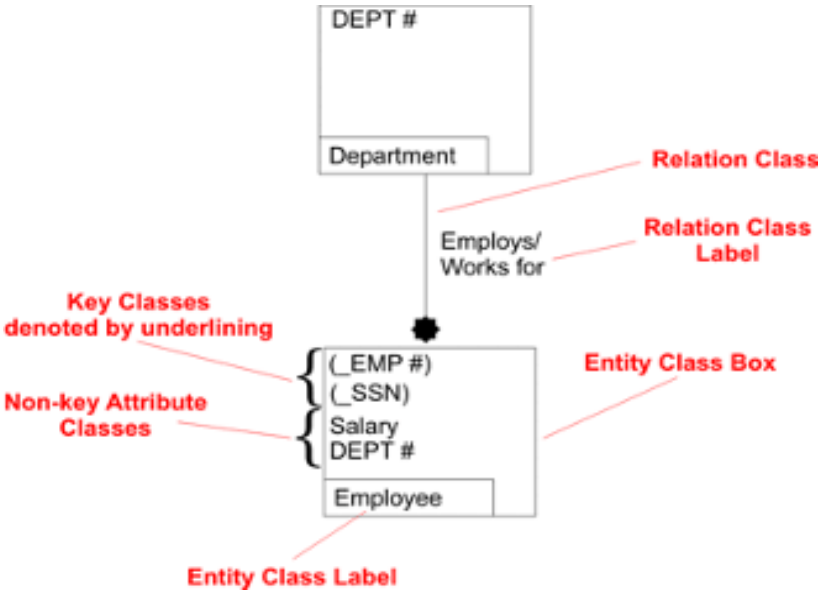
Permodelan Informasi

Spefikasi Informasi dan Data

- Arsitektur informasi adalah deskripsi rancangan terstruktur dari lingkungan informasi.
- Spesifikasi fungsi menggunakan IDEF0
- Spesifikasi Informasi menggunakan IDEF1
- Spesifikasi data menggunakan IDEF1x



Pemodelan Fungsi (IDEF0)



Pemodelan Informasi (IDEF1)

Konversi antara IDEF0 dan IDEF1

IDEF0	IDEF1
Input	Entity class
Output	Domain class
Control	Attribute class
Relationship : Output-Input	Memiliki atau bagian dari
Relationship : Output control	1..1, 1…*, *..*

Definisi dan Fungsi

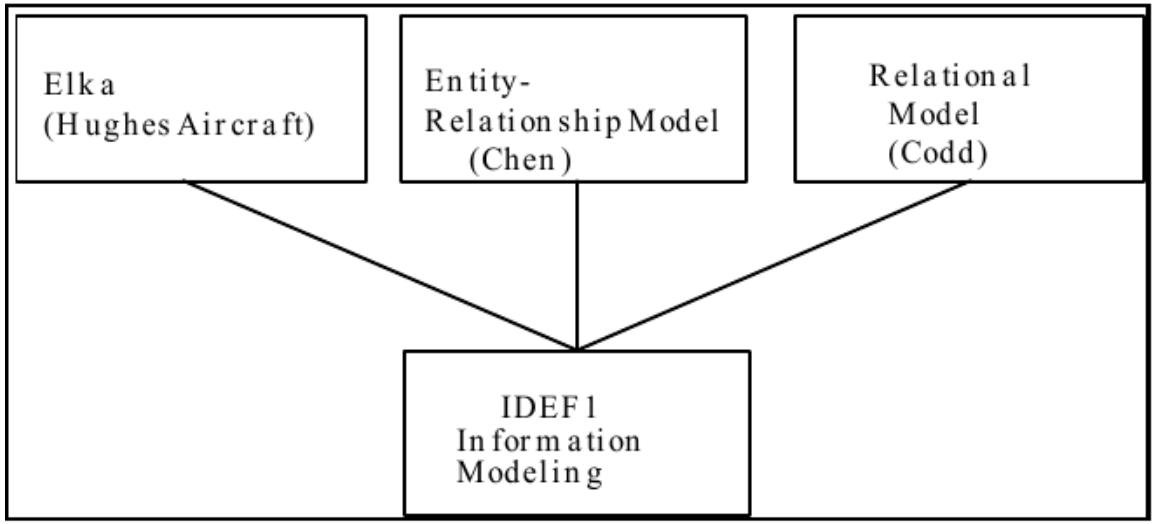
- DEFINISI :
  - Metode untuk menganalisis dan mengkomunikasikan struktur dan semantik informasi dalam suatu sistem.
- FUNGSI :
  - Identifikasi informasi yang dikelola sistem dunia nyata.
  - Identifikasi aturan untuk mengelola informasi.
  - Identifikasi kekurangan dalam manajemen informasi saat ini.
  - Tentukan informasi apa yang akan dikelola dalam implementasi.

KEKUATAN

- Memerlukan partisipasi aktif dari pengguna informasi.
- Metode efektif untuk mendokumentasikan persyaratan informasi dari suatu perusahaan.
- Memberlakukan modularitas yang menghilangkan ketidaklengkapan, ketidakkonsistenan, dan ketidaktepatan yang ditemukan dalam proses pemodelan.

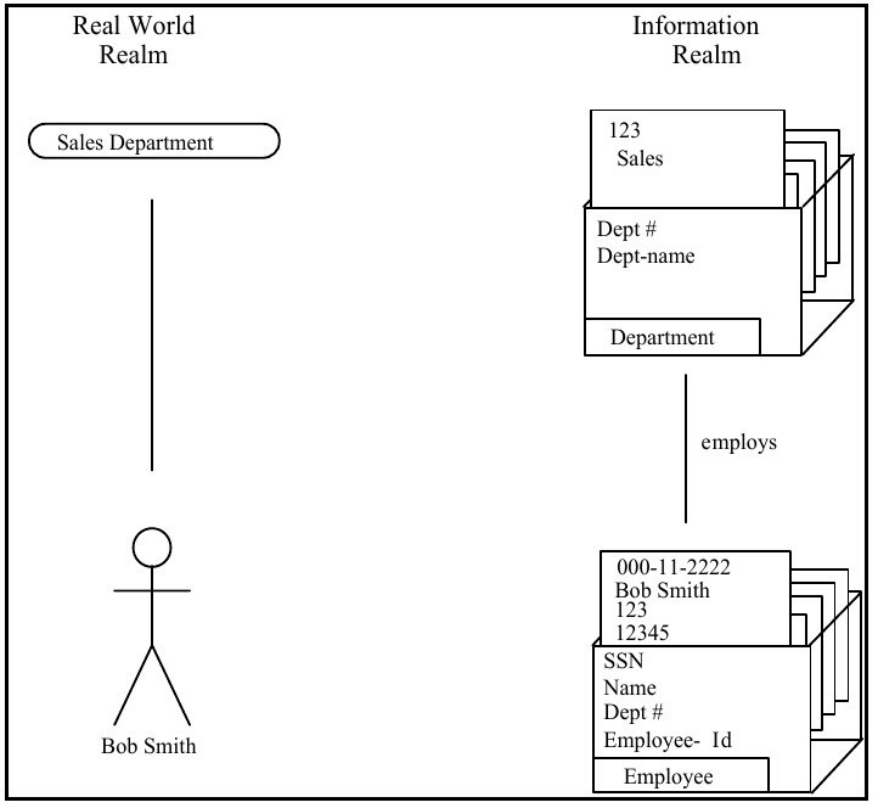
IDEF1

- Obyek di dunia nyata
- Asosiasi fisik atau abstrak yang mengaitkan dengan obyek dunia nyata
- Informasi yang dikelola terkait obyek dunia nyata
- Struktur data yang digunakan untuk mereprsentasikan informasi yang dikumpulkan, diterapkan dan pengelolaan informasi tersebu



Konsep IDEF1

- There are two important realms for modelers to consider in determining information requirements.
  - The first realm is **the real world as perceived by people** in an organization. It is comprised of the physical and conceptual objects (e.g., people, places, things, ideas, etc.), the properties of those objects, and the relations associated with those objects.
  - The second realm is the information realm. It includes information images of those objects found in the real-world. An information image is not the real-world object, but **the information collected, stored, and managed about real-world objects**.
- IDEF1 is designed to assist in discovering, organizing, and documenting this information image, and thus is restricted to the information realm.



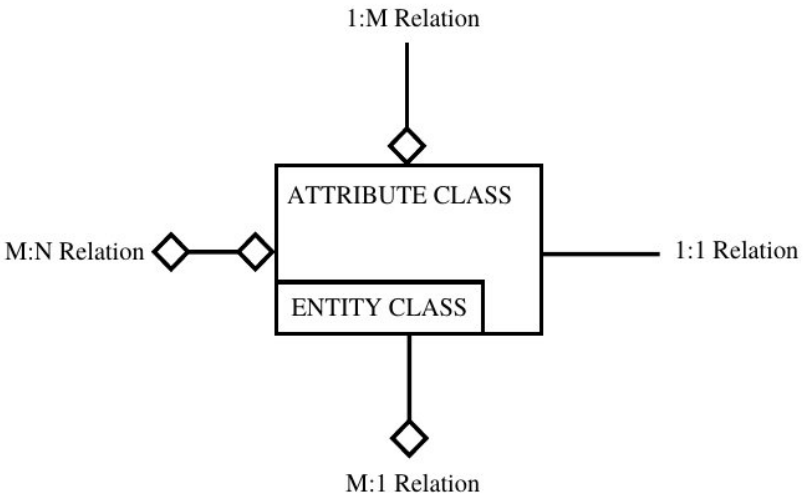
Pemanfaatan IDEF1

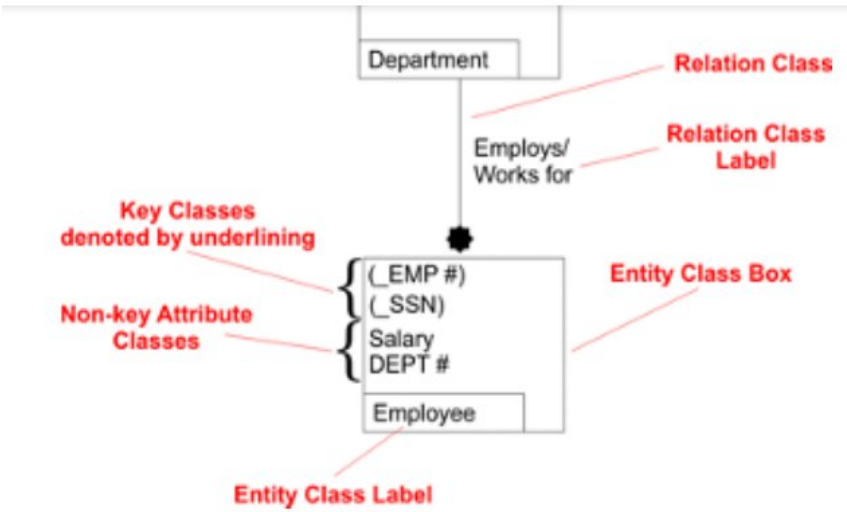
- IDEF1 was designed as a method for both analysis and communication in the establishment of requirements. IDEF1 is generally used to
  - identify what information is currently managed in the organization,
  - determine which of the problems identified during the needs analysis are caused by lack of management of appropriate information, and
  - specify what information will be managed in the TO-BE implementation.
- IDEF1 captures the information that exists about objects within the scope of an enterprise. The IDEF1 perspective of an information system includes not only the automated system components, but also non- automated objects such as people, filing cabinets, telephones, etc.
- IDEF1 was designed as a method for organizations to analyze and clearly state their information resource management needs and requirements.

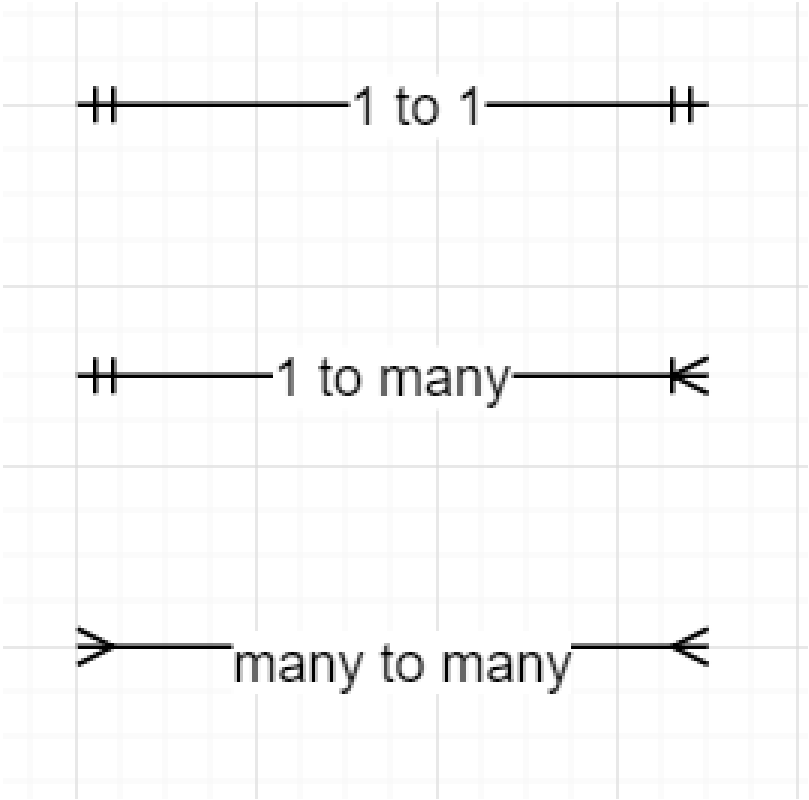
Penggunaan IDEF1

- Rather than a database design method, IDEF1 is an analysis method used to identify the following:
  - Information collected, stored, and managed by the enterprise.
  - Rules governing the management of information.
  - Logical relationships within the enterprise re ected in the information.
  - Problems resulting from the lack of good information management.

Notasi


1:M Relation
ATTRIBUTE CLASS
M:N Relation
1:1 Relation
ENTITY CLASS
M:1 Relation

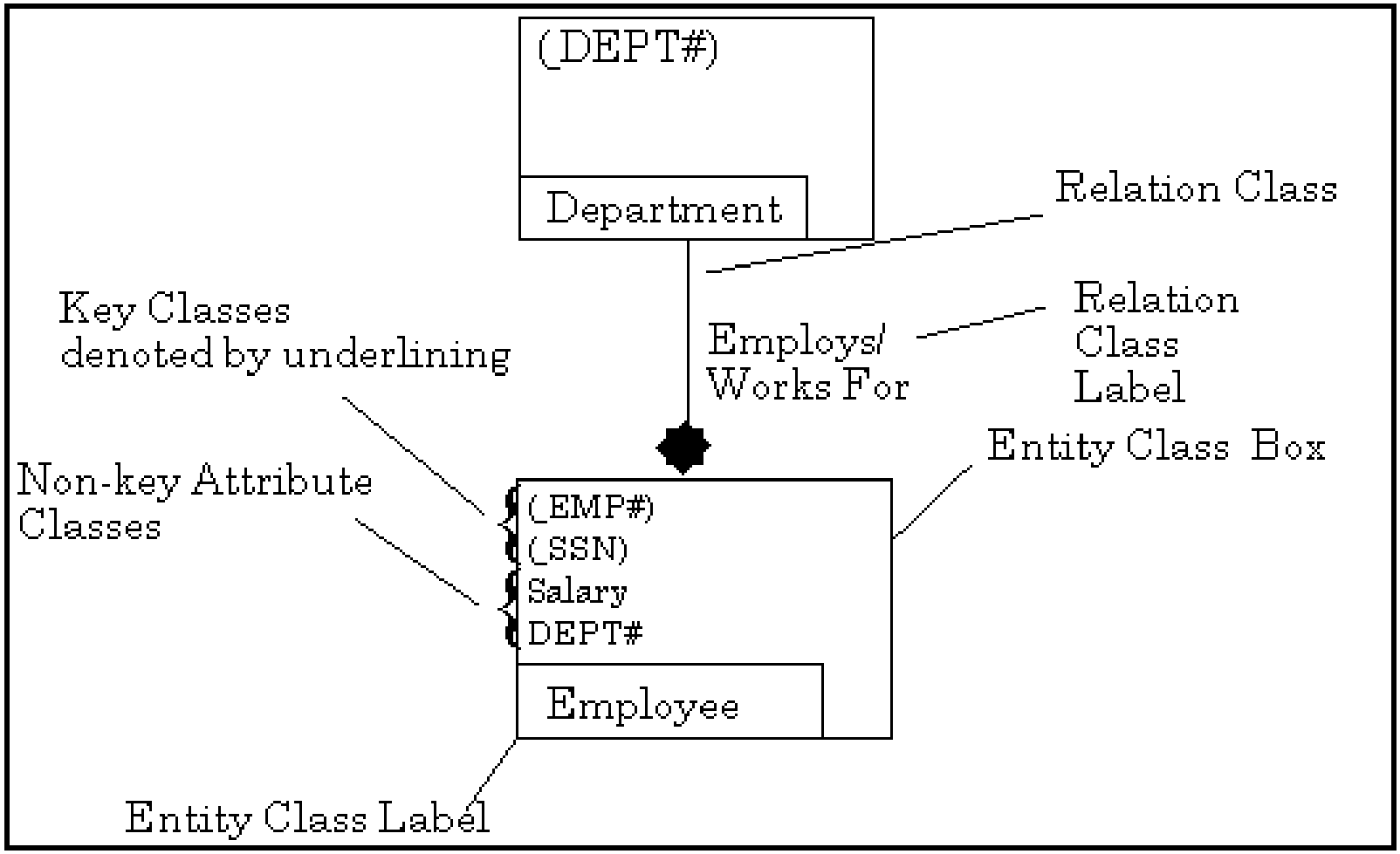

Department
Relation Class
Employs/
Works for
Relation Class
Label
Key Classes
denoted by underlining
(_EMP #)
(_SSN)
Salary
DEPT #
Entity Class Box
Non-key Attribute
Classes
Employee
Entity Class Label


1 to 1
1 to many
many to many

KONSEP UTAMA

- **Entitas**: Mewakili informasi yang dikelola dalam sistem tentang objek dunia nyata
- **Atribut**:
  - Kunci
  - Non-kunci
- **Relasi** : Asosiasi antar entitas

IDEF1- Information Modeling Method



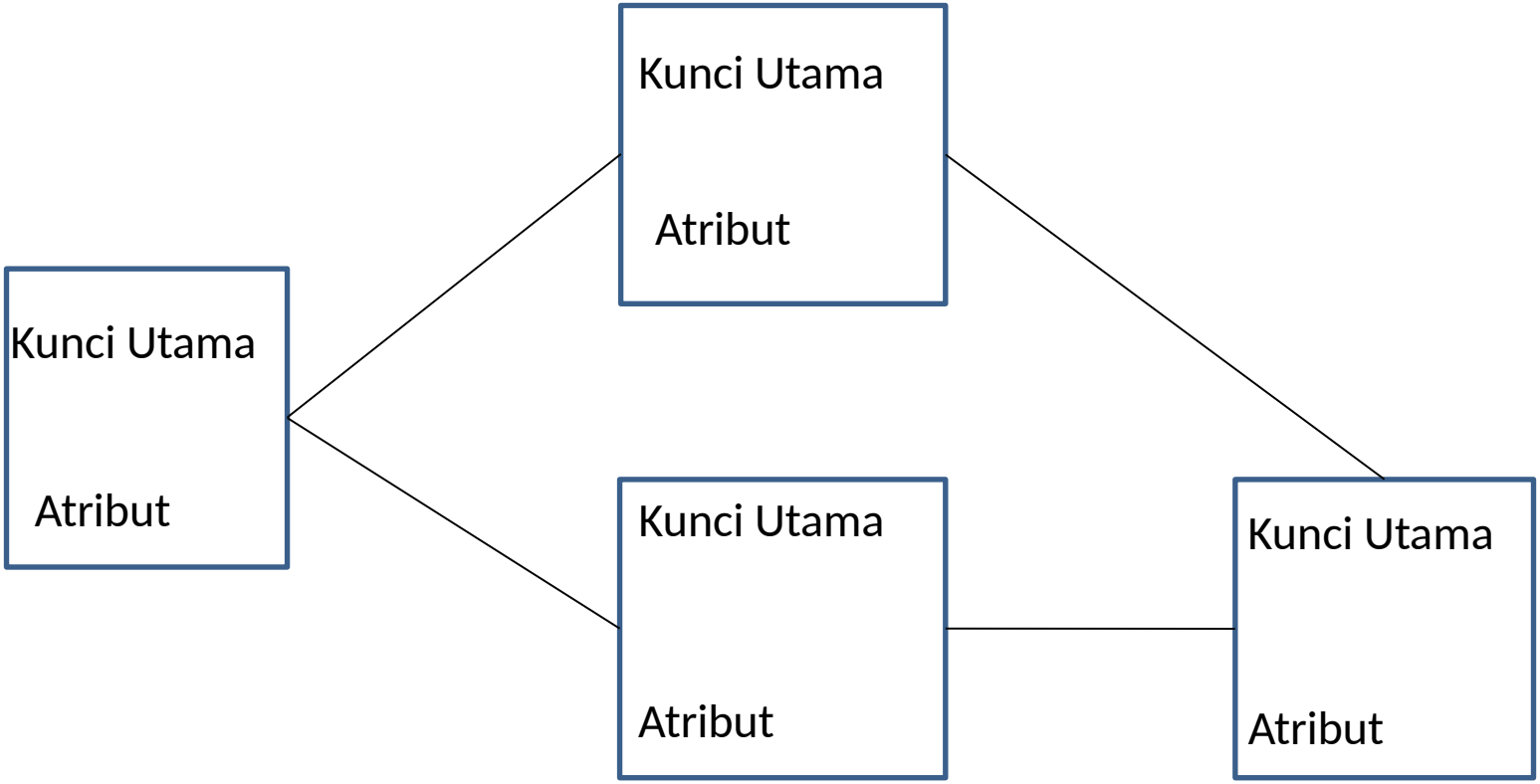
Entitas

- An IDEF1 entity represents the information maintained in a specific organization about physical or conceptual objects. An IDEF1 entity class refers to a collection of entities or the class of information kept about objects in the real-world.
- There are two basic concepts that distinguish entities:
  - **They are persistent.** The organization expends resources to observe, encode, record, organize, and store the existence of individual entities.
  - **They can be individuate**d. They can be identified uniquely from other entities.
-

Attribute

- Entities have characteristic attributes associated with them.
- Attributes record values of properties of the real- world objects. The term attribute class refers to the set of attribute-value pairs formed by grouping the name of the attribute and the values of that attribute for individual entity class members (entities).
- A collection of one or more attribute classes which distinguishes one member of an entity class from another is called a key class.
- A relation in IDEF1 is an association between two individual information images. The existence of a relation is discovered or verified by noting that the attribute classes of one entity class contain the attribute classes of the key class of the referenced entity class member.

Struktur IDEF1 suatu sistem



Menetapkan informasi

- Data Barang
- Supplier
- List Pemesanan Barang
- Faktur Pembayaran
- Laporan Pemesanan Barang
- Permintaan Tetap
- Permintaan Darurat
- Pemesanan Tetap
- Pemesanan Darurat
- Laporan Pengeluaran Barang

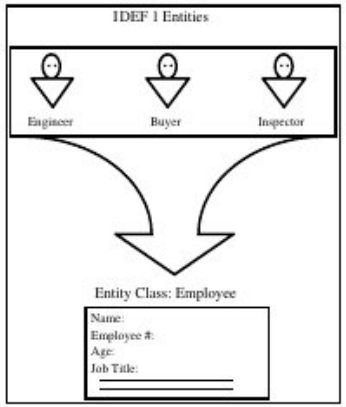
Matriks Hubungan Entitas Dalam IDEF1

	Tabel 1	Tabel 2	Tabel 3	Tabel 4
Tabel 1		x		x
Tabel 2	x		x	
Tabel 3		x		
Tabel 4	x			

Matriks Relasi

	Data Barang	Supplier	List Pemesanan Barang	Faktur Pembayaran	Laporan Penerimaan Barang	Permintaan Tetap	Permintaan Darurat	Pemesanan Tetap	Pemesanan Darurat	Laporan Pengeluaran Barang
Data Barang		X			X	X	X	X	X	
Supplier	X			X						
List Pemesanan Barang								X	X	
Faktur Pembayaran		X			X					
Laporan Pemesanan Barang	X			X						
Permintaan Tetap	X									X
Permintaan Darurat	X		X							X
Pemesanan Tetap	X		X							
Pemesanan Darurat										
Laporan Pengeluaran Barang						X	X			

Tahapan menyusun IDEF1



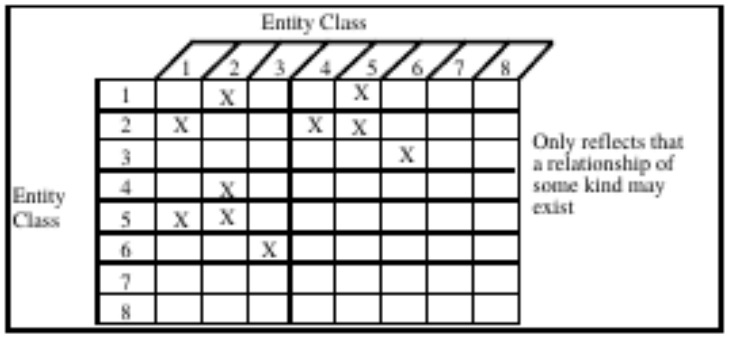
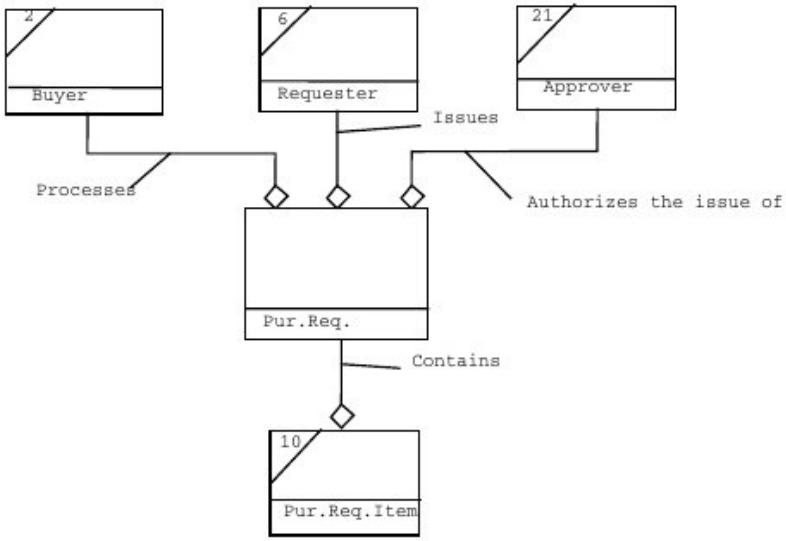
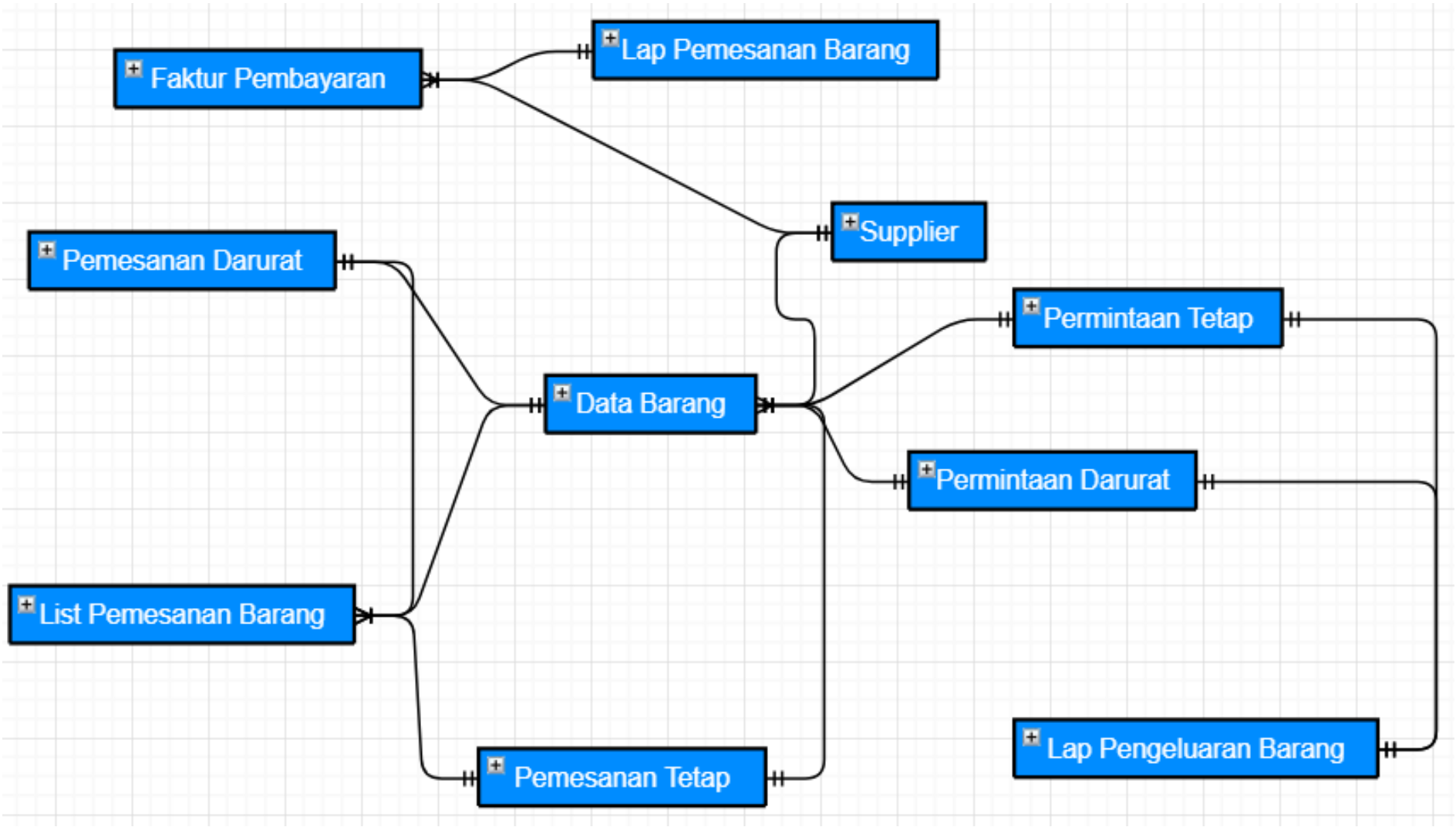


Figure 3-6. Relation Matrix

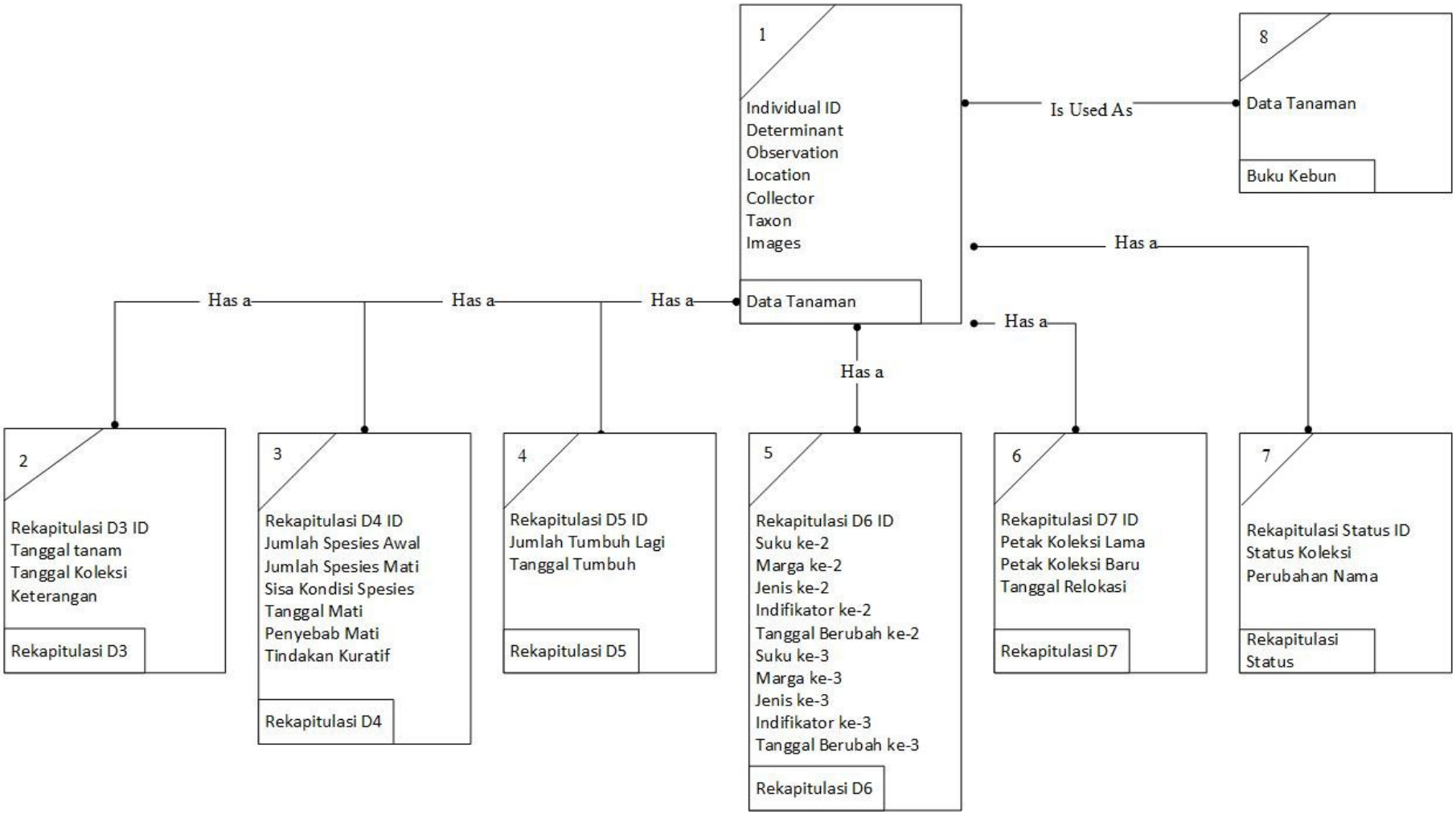


1) Inventarisasi Entitas → Informasi yang ada di dunia nyata
2) Susun matriks relasi antara Entitas
3) Identifikasi Attribute
4) Identifikasi Attribute2 yang KEY dan Non Key
5) Gamnbarkan IDEF 1

MODEL IDEF1

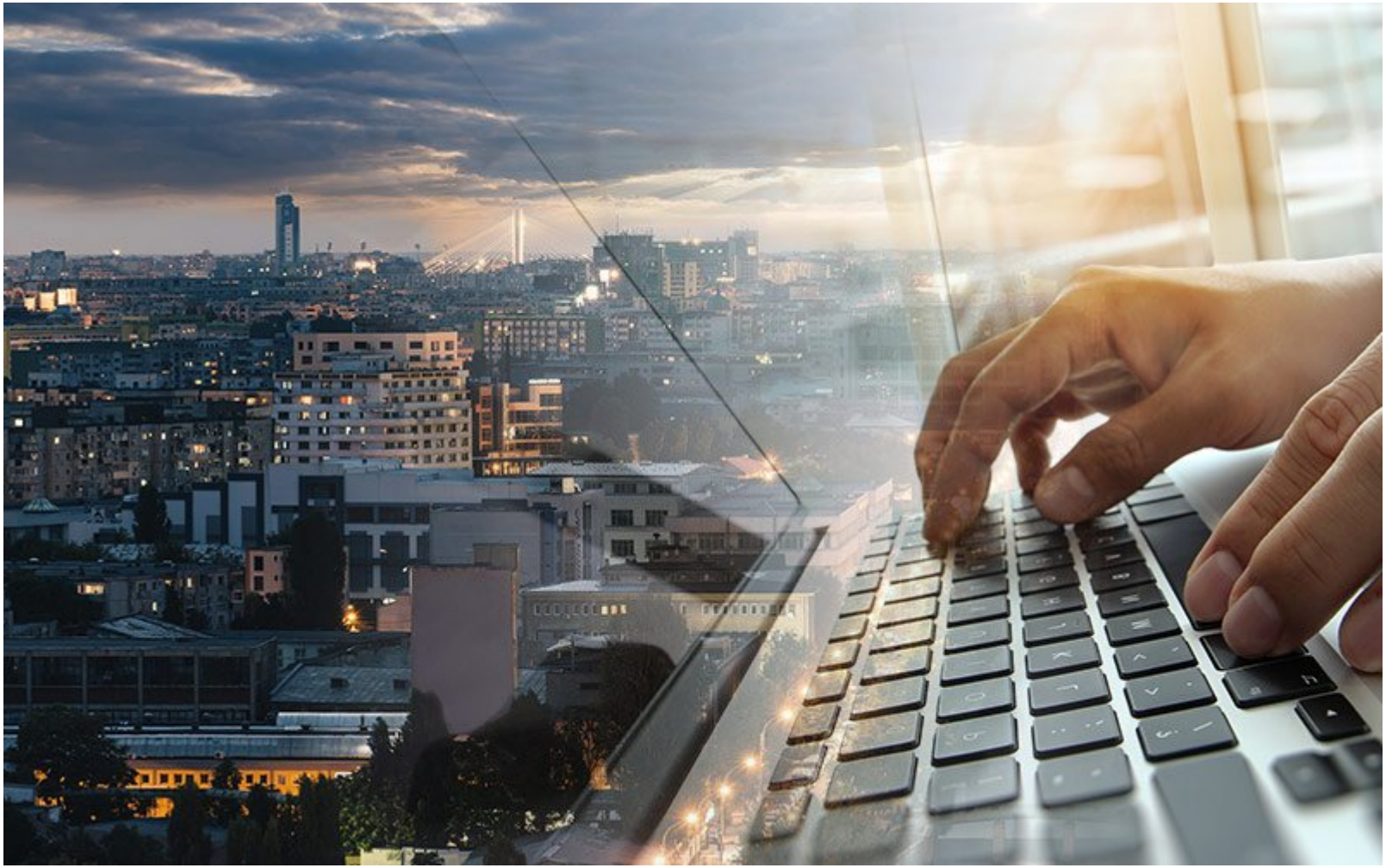

Faktur Pembayaran
Lap Pemesanan Barang
Supplier
Pemesanan Darurat
Permintaan Tetap
Data Barang
Permintaan Darurat
List Pemesanan Barang
Lap Pengeluaran Barang
Pemesanan Tetap

IDEF 1 – Information Modelling



Tugas

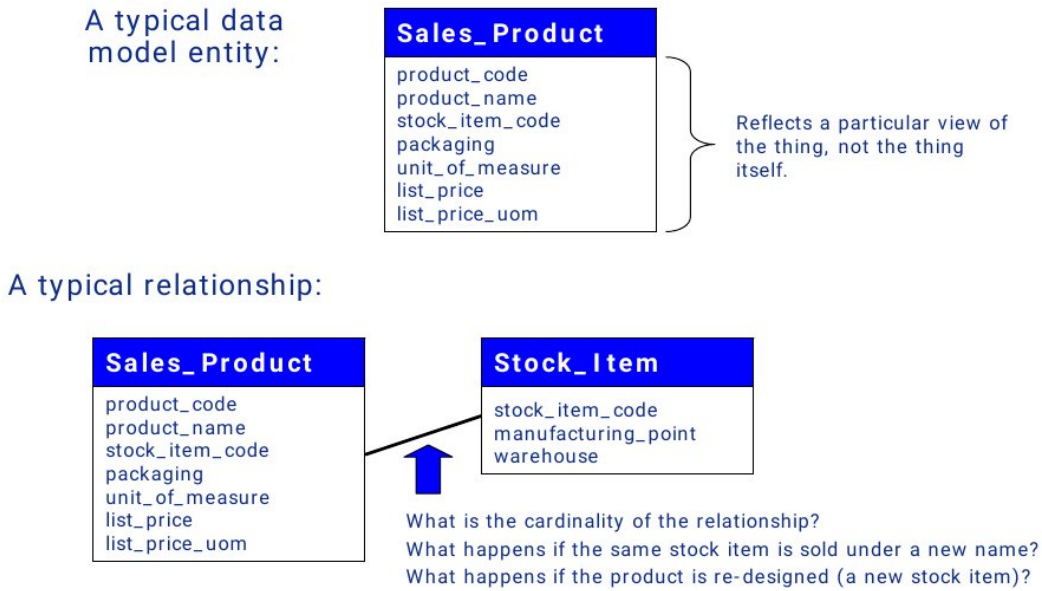
- Tentukan entitas dari IDEF1
  - Identifikasi real-world realm
  - Identifikasi information realm (informasi yang menggambarkan obyek tersebut), dengan cara menginventaris
    - informasi yang masuk ke sistem (formulir, screen, kartu isian)
    - Informasi yang dihasilkan sistem (report keluaran, kartu cetakan)
- Susun matrix relasi antar entitas
- Melihat attribute pada obyek tersebut, mana yang key mana yang non key
- Gambarkan IDEF1



Permodelan Data

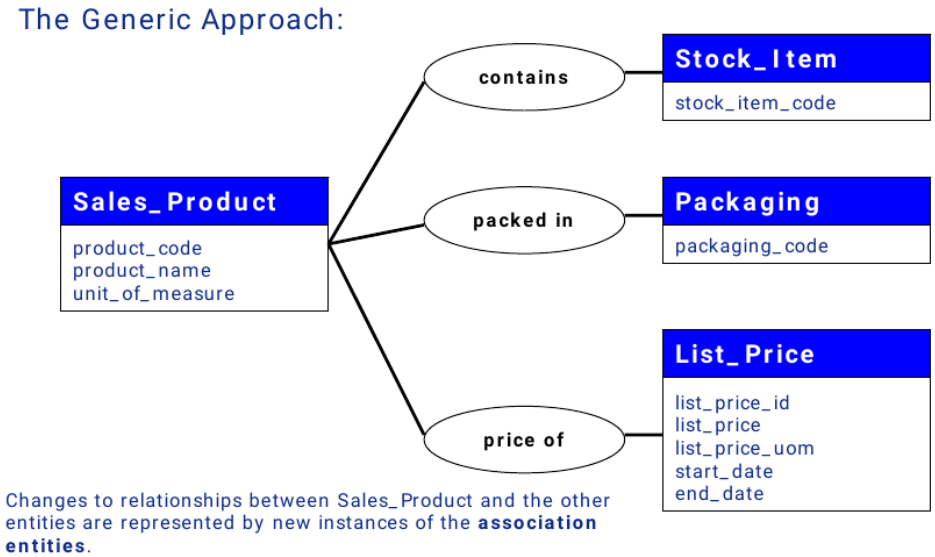
Data Model

CONVENTIONAL DATA MODEL



- Garis tidak memiliki semantik khusus
- Membatasi pada model RDBMS
- Relation → reference
- Dikenal dengan ERD,

GENERIC DATA MODEL



- Garis memilki semantik yang berbeda
- Lebih tidak terikat pada model data
- Model ontologi, frame, semantic net
- Lebih tepat memotret beragam data

Data Models

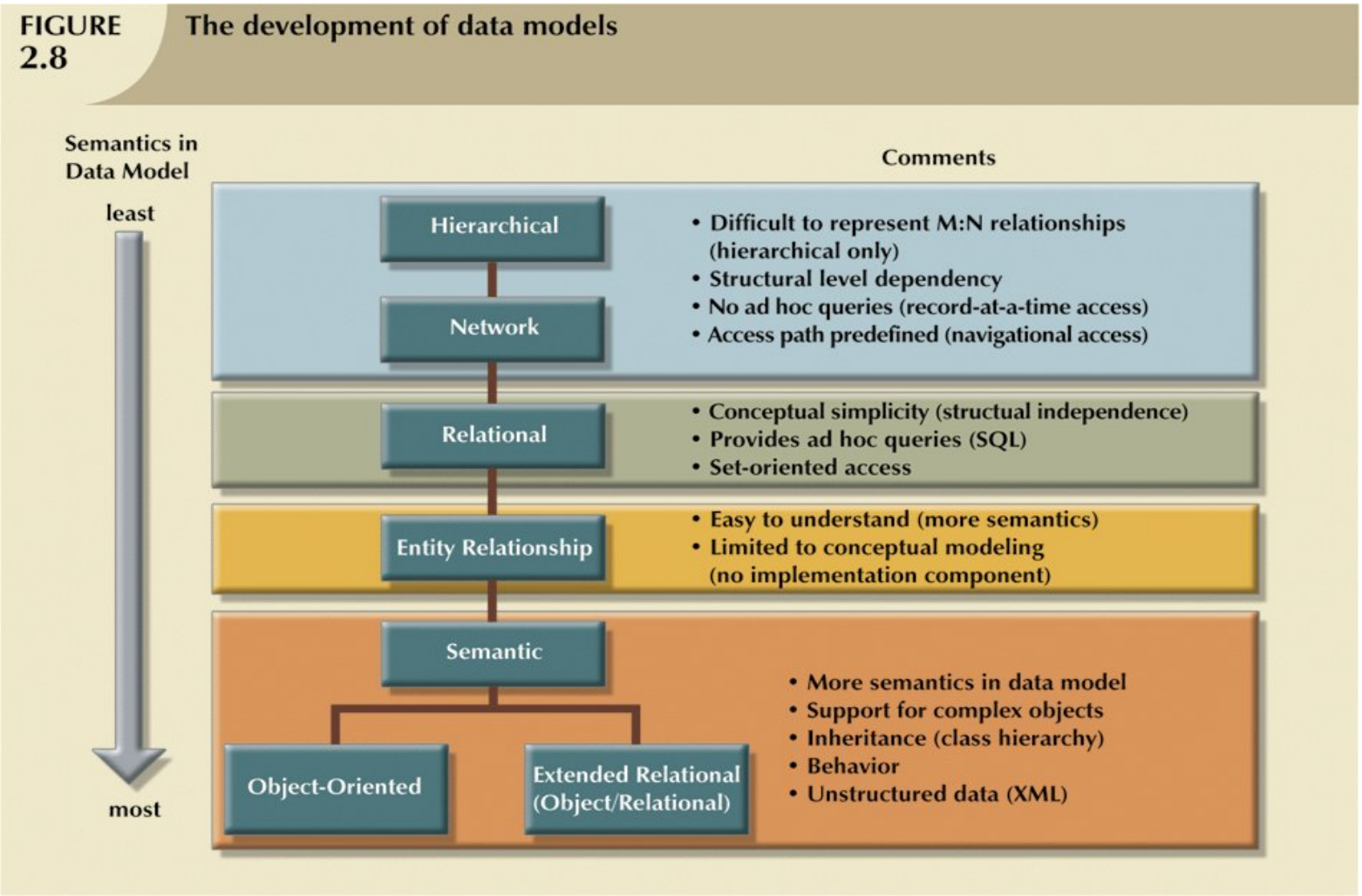


FIGURE 2.8
The development of data models
Semantics in Data Model
least
most
Comments
Hierarchical
Network
• Difficult to represent M:N relationships (hierarchical only)
• Structural level dependency
• No ad hoc queries (record-at-a-time access)
• Access path predefined (navigational access)
Relational
• Conceptual simplicity (structual independence)
• Provides ad hoc queries (SQL)
• Set-oriented access
Entity Relationship
• Easy to understand (more semantics)
• Limited to conceptual modeling (no implementation component)
Semantic
Object-Oriented
Extended Relational (Object/Relational)
• More semantics in data model
• Support for complex objects
• Inheritance (class hierarchy)
• Behavior
• Unstructured data (XML)

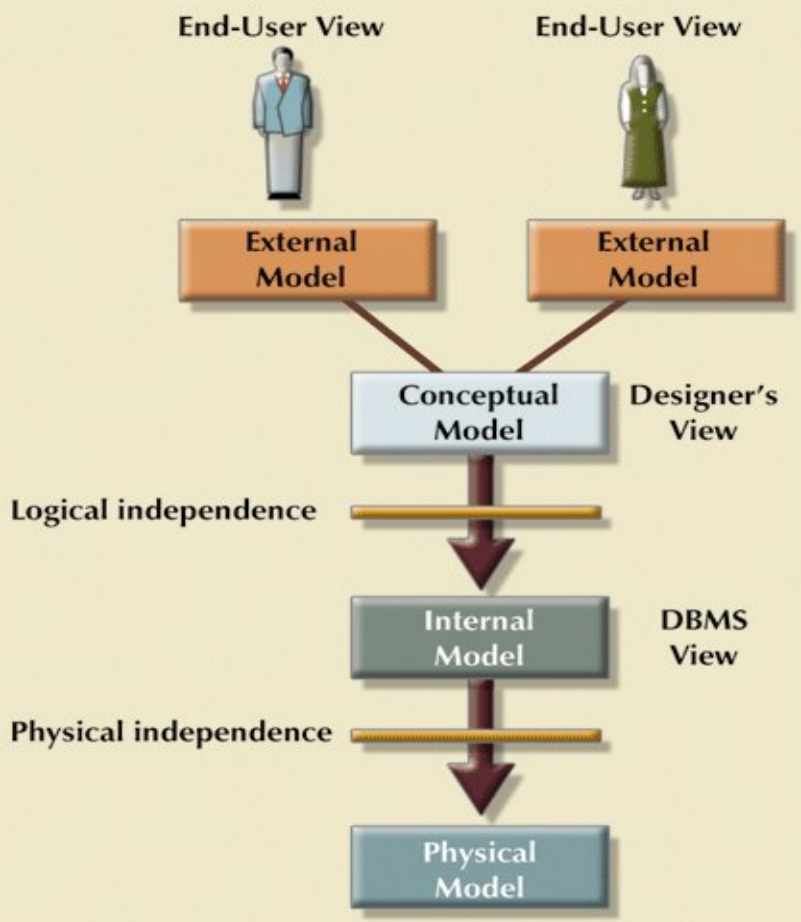
Evolution of Major Data Models

TABLE 2.1 Evolution of Major Data Models

GENERATION	TIME	MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File system	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and network	IMS ADABAS IDS-II	Early database systems Navigational access
Third	Mid-1970s to present	Relational	DB2 Oracle MS SQL-Server	Conceptual simplicity Entity relationship (ER) modeling support for relational data modeling
Fourth	Mid-1980s to present	Object-oriented Extended Relational	Versant VFS/FastObjects Objectivity/DB DB/2 UDB Oracle 10g	Support complex data Extended relational products support objects and data warehousing Web databases become common
Next Generation	Present to future	XML	dbXML Tamino DB2 UDB Oracle 10g MS SQL Server	Organization and management of unstructured data Relational and object models add support for XML documents

Tingkat Abstraksi Data

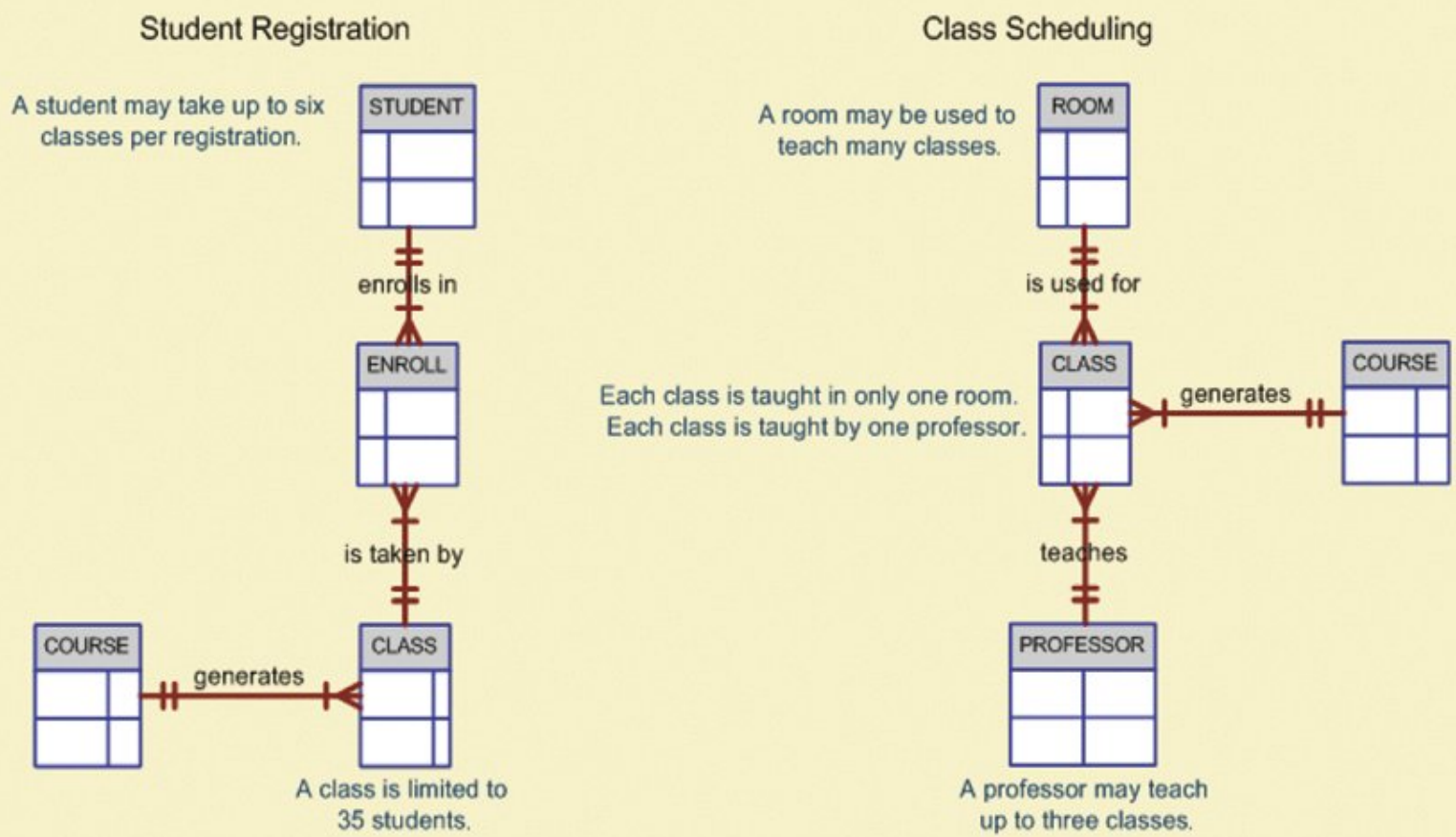
FIGURE 2.9 Data abstraction levels



Degree of Abstraction		Characteristics
High	ER	Hardware-independent Software-independent
Medium	Relational Object-Oriented	Hardware-independent Software-dependent
Low	Network Hierarchical	Hardware-dependent Software-dependent

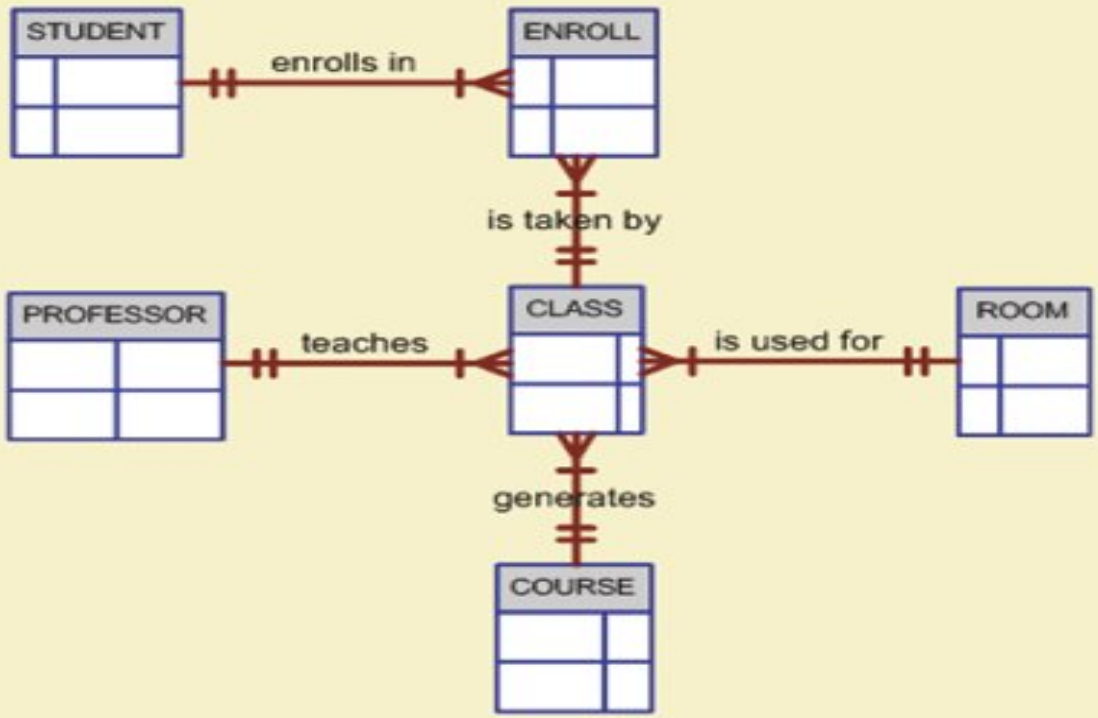
The External Model

FIGURE 2.10 External models for Tiny College



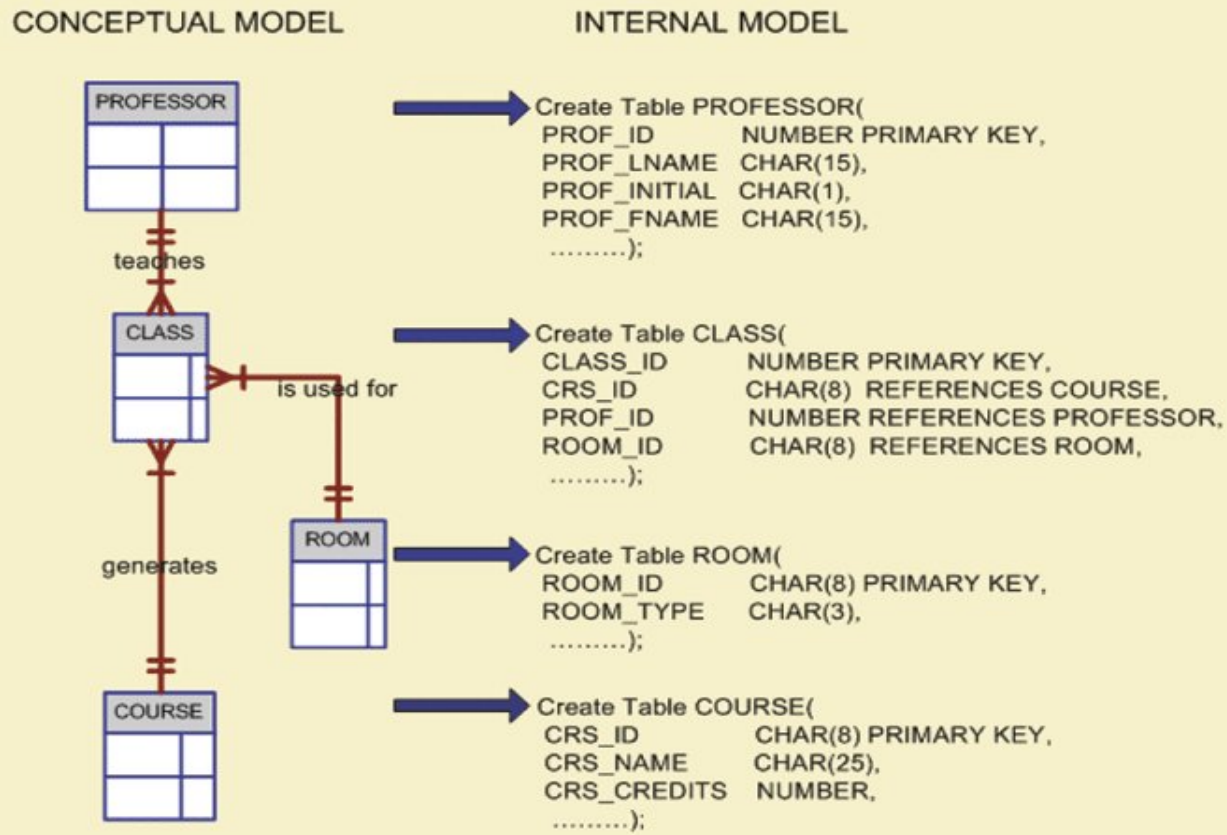
The Conceptual Model (continued)

FIGURE 2.11 Conceptual model for Tiny College



The Internal Model

FIGURE 2.12 An internal model for Tiny College



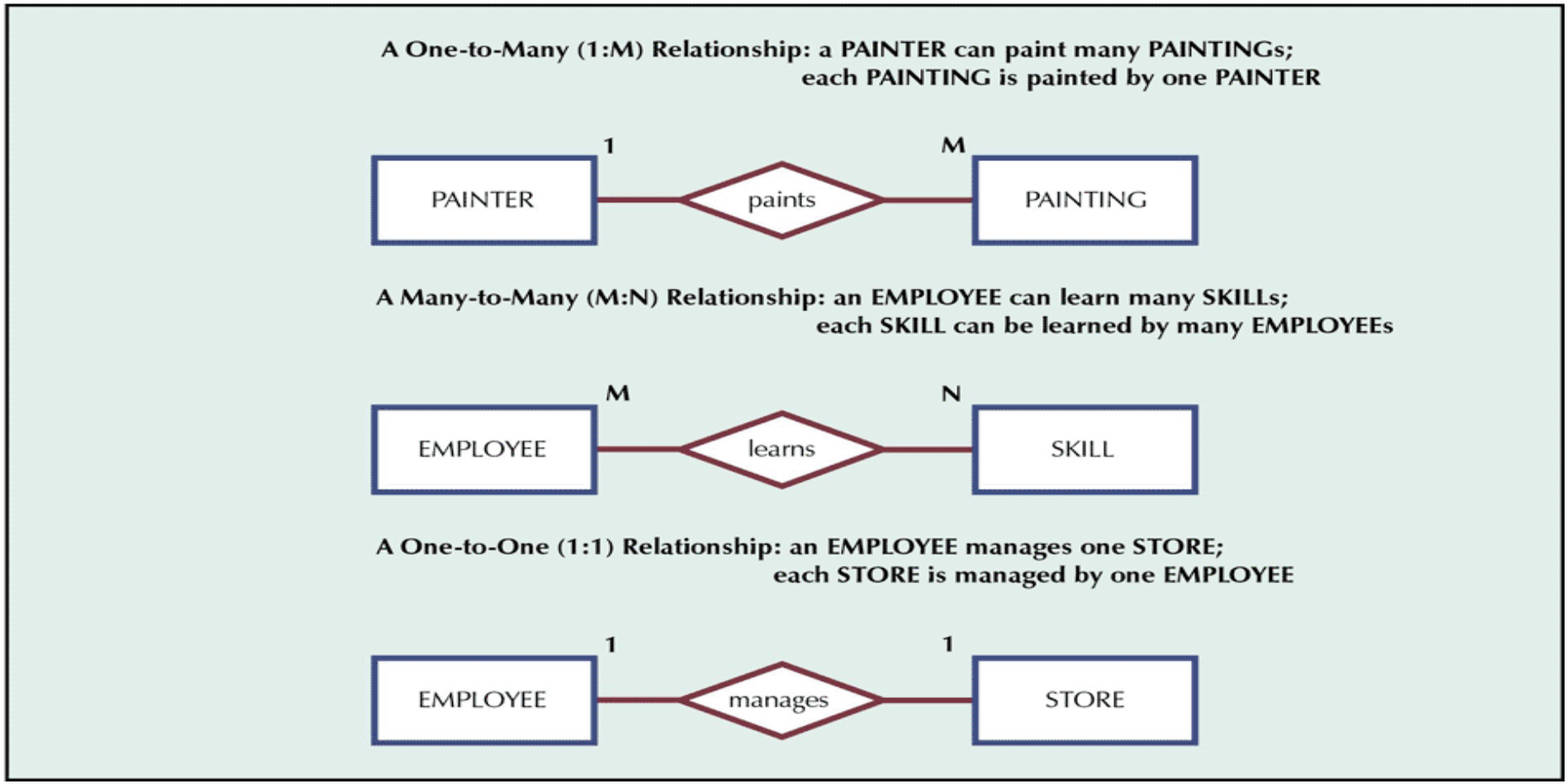
The Physical Model (continued)

TABLE 2.3 Levels of Data Abstraction

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual	↕	Global view of data (independent of database model)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

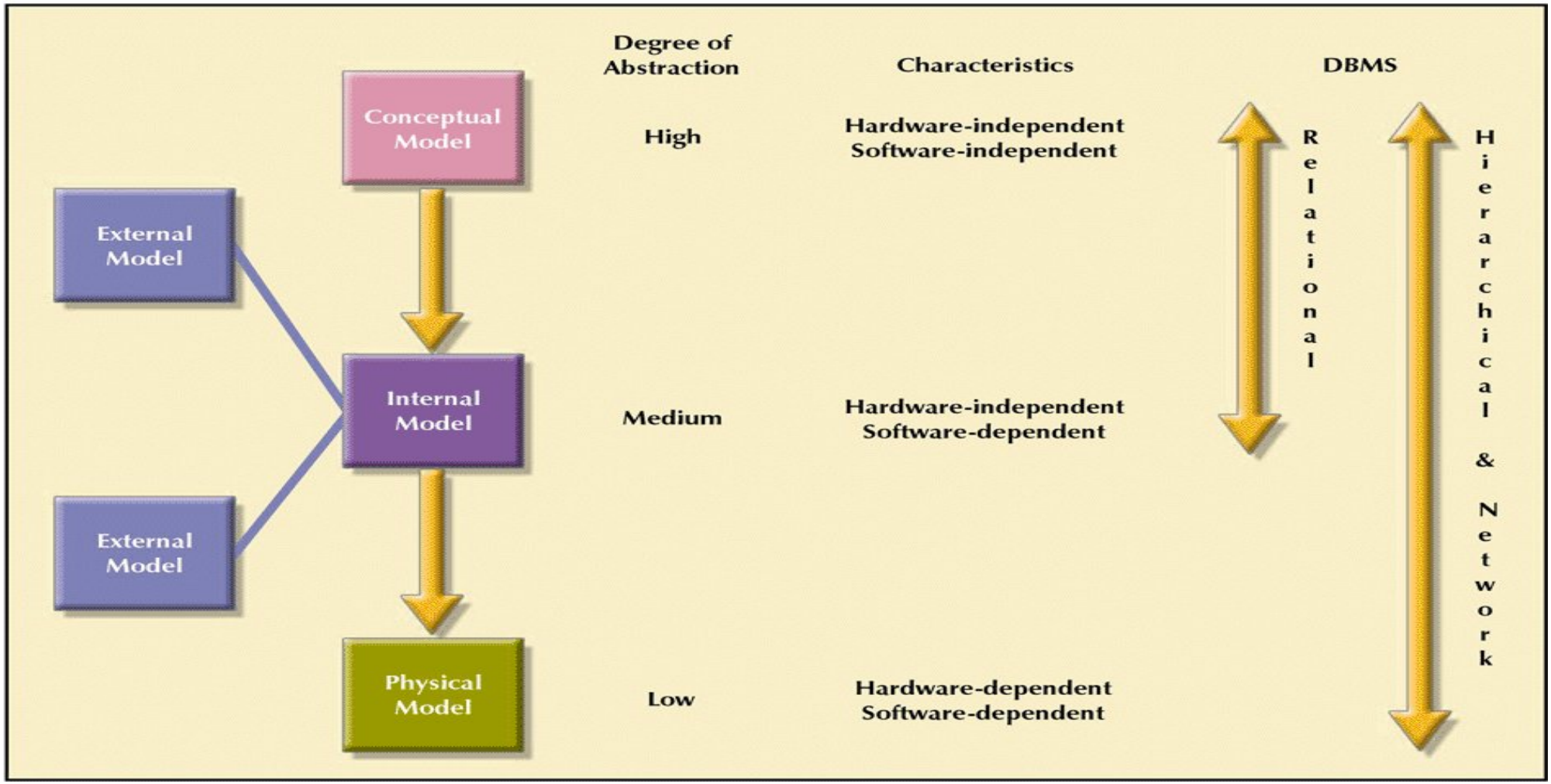
Relationships: The Basic Chen ERD

FIGURE 2.6 RELATIONSHIPS: THE BASIC CHEN ERD



Data Abstraction Levels

FIGURE 2.10 DATA ABSTRACTION LEVELS



Levels of Data Abstraction

TABLE 2.2 LEVELS OF DATA ABSTRACTION

MODEL	DEGREE OF ABSTRACTION	DATA MODEL	FOCUS	INDEPENDENT OF
Conceptual	High	Entity	Global	Hardware and software
External		ER components	Subset	Hardware
Internal		Relational and others	Global	Hardware
Physical	Low	Physical storage methods	N/A	Neither hardware nor software

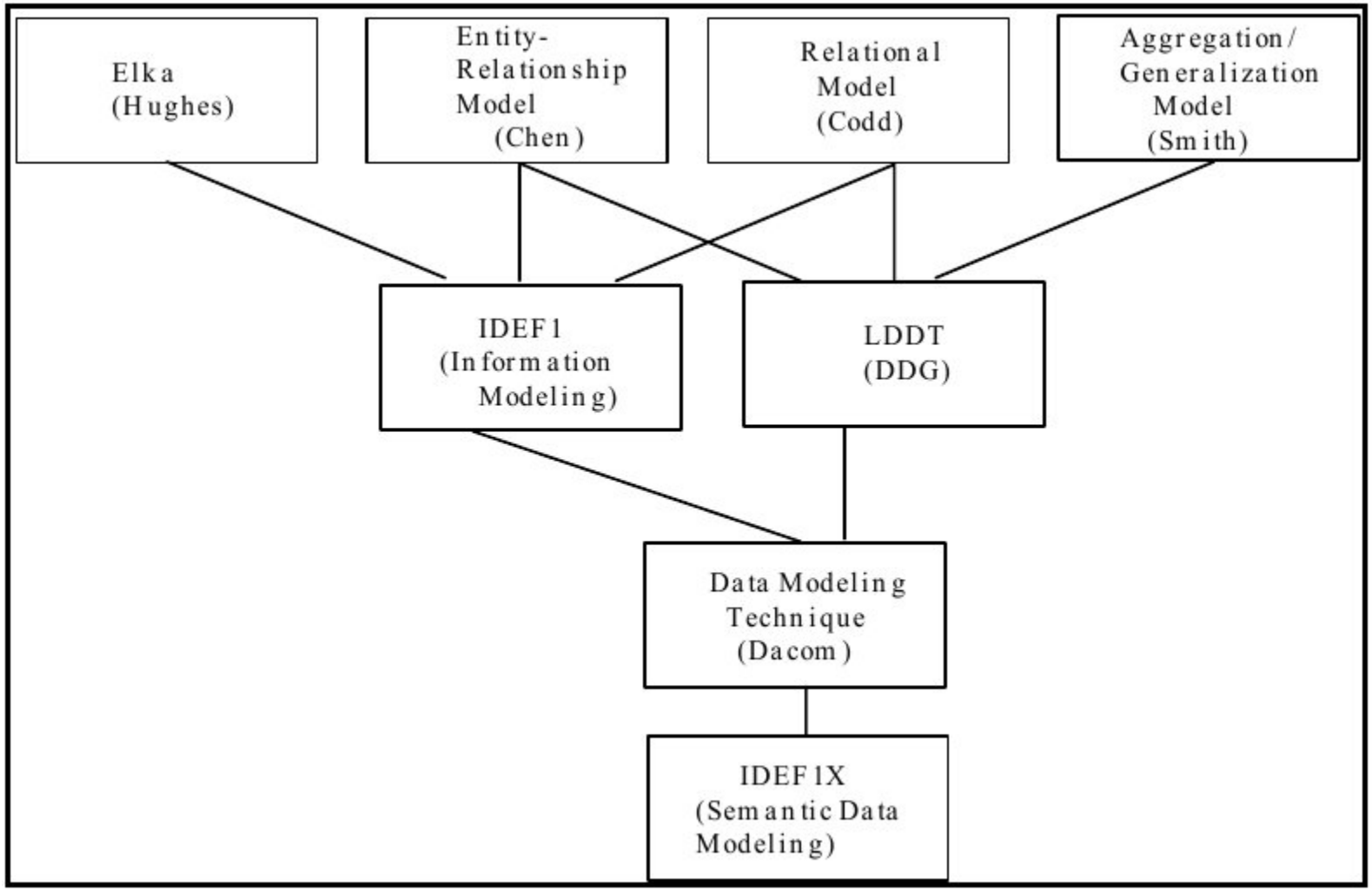
What is Data Modeling?

- Conceptual (sometimes called Logical)
  - Gather data requirements for business functions being analyzed
  - Think "WHAT???" – "What data necessary to do the business function"
  - It's all about the business, do not impose "Information System" terms or techniques
    - No such business things as tables, keys, indexes and the like
    - Probably best done by business analysts in sync with BPM
    - In some shops, data modelers can bridge the business/IT gap
- Physical
  - Translate logical model into an implementation plan for a database
  - Think "HOW???" – "How to construct database objects to realize this"
  - It's all about the target database or other technology being adopted
  - Now it's time to think about tables, keys, indexes and the like
  - Probably best done by data architects, data modelers, and DBA's

KEKUATAN

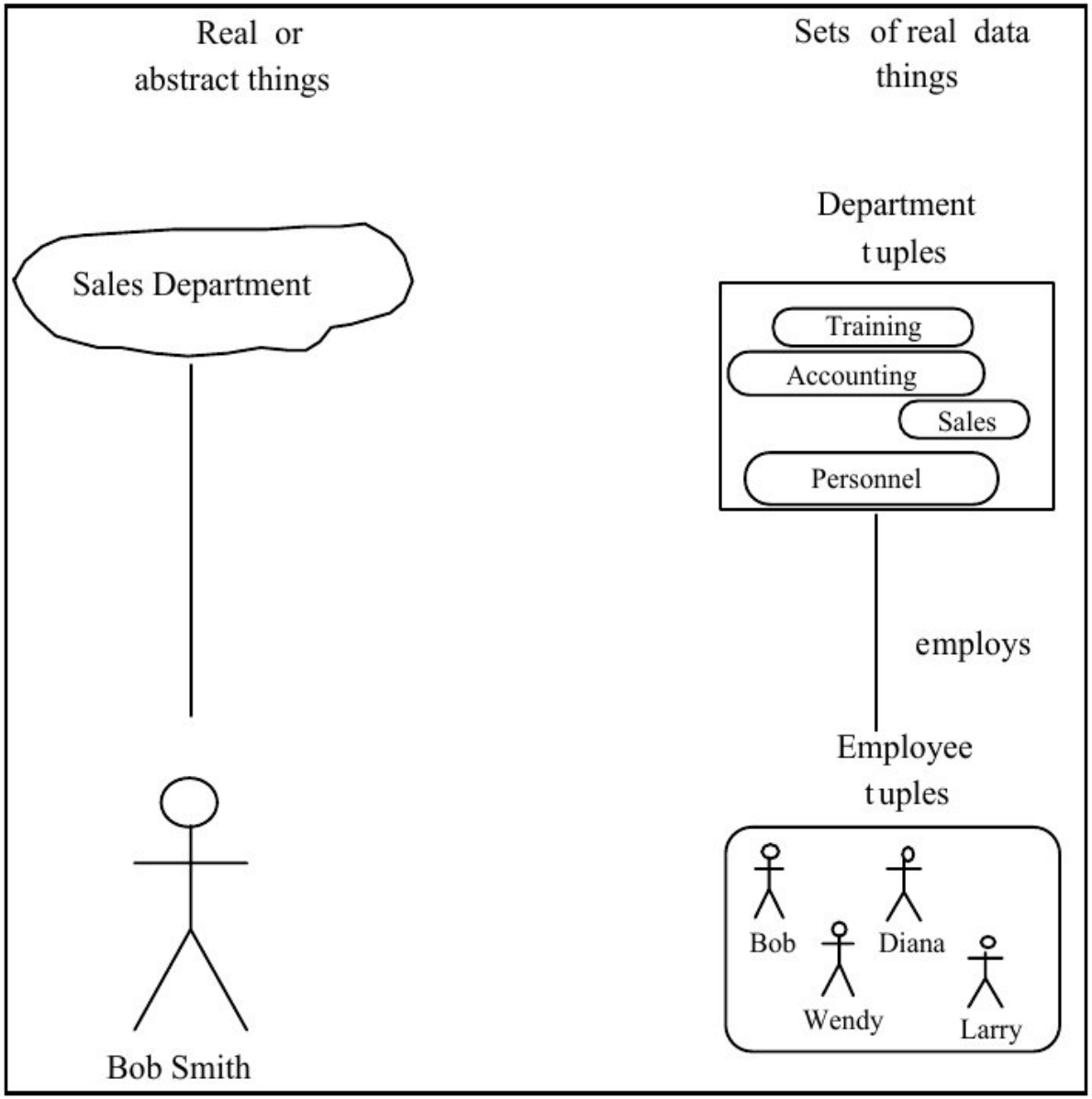
- Kekuatan :
  - Alat kuat untuk pemodelan data.
  - IDEF1X tidak memiliki banyak varian.
  - Menggambarkan aturan yang mengatur manajemen informasi.
  - Digunakan untuk memvalidasi konsep IDEFO.
  - Membantu menemukan penyebab masalah.
- Kelemahan
  - Pemodel harus berpengalaman untuk menciptakan model yang baik.
  - Tidak cocok untuk digunakan sebagai alat analisis.

IDEF1x – Data modelling



Konsep IDEF1x

- Menjembatani antara informasi di dunia nyata dengan model data di sistem
- Mendisain spesifikasi data
- Memodelkan obyek model bukan relasi



IDEF1X

- IDEF1X is a method for designing relational databases with a syntax designed to support the **semantic constructs necessary** in developing a conceptual schema.
- A conceptual schema is a single integrated definition of the enterprise data that is unbiased toward any single application and independent of its access and physical storage.
- Because it is a design method, IDEF1X is not particularly suited to serve as an AS-IS analysis tool, although it is often used in that capacity as an alternative to IDEF1.
- IDEF1X is most useful for **logical database design** after the information requirements are known and the decision to implement a relational database has been made. Hence, the IDEF1X system perspective is focused on the **actual data elements in a relational database**.
- If the target system is **not a relational system**, for example, an object-oriented system, **IDEF1X is not the best method**.

Perbedaan IDEF dan IDEF1X

- Although the terminology between IDEF1 and IDEF1X is very similar, there are fundamental differences in the theoretical foundations and concepts of the two methods.
- An entity in **IDEF1X refers to a collection or set of similar data instances that can be individually distinguished** from one another. Individual members of the set are called entity instances.
- Thus, a box in IDEF1X represents a set of data items in the real-world realm. An attribute is a slot value associated with each individual member of the set.
- The relationship that exists between individual members of these sets is given a name. In this case, this relation establishes a referential integrity constraint.

IDEF1x- Data Modeling Method

- IDEF1X is a method for designing relational databases.
- IDEF1X Model: Graphical and Textual depiction of “What must I know to do what I do?”
- IDEF1X is most useful for logical database design *after* the information requirements are known and the decision to implement a relational database has been made.
- IDEF1X was released as a standard for Data Modeling by the Computer Systems Laboratory of the National Institute of Standards and Technology. (1993)

IDEF1x- Data Modeling Method

- EXAMPLE (ref:[2])

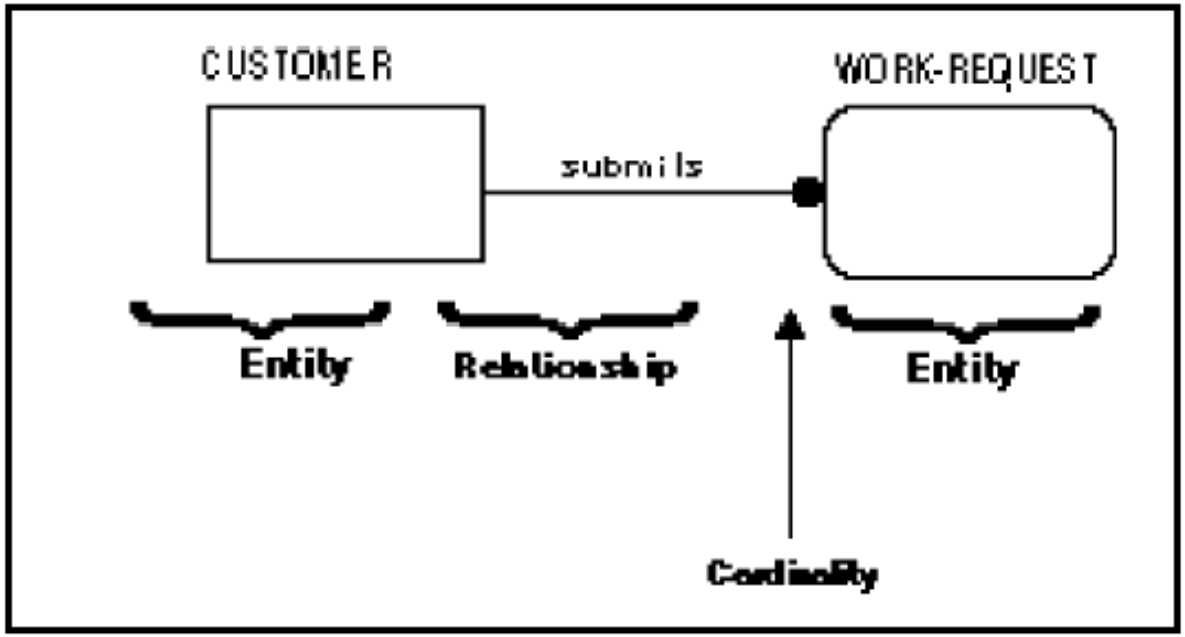


Figure 3-3. Basic Data Model

IDEF1x- Data Modeling Method

- EXAMPLE (ref:[2])

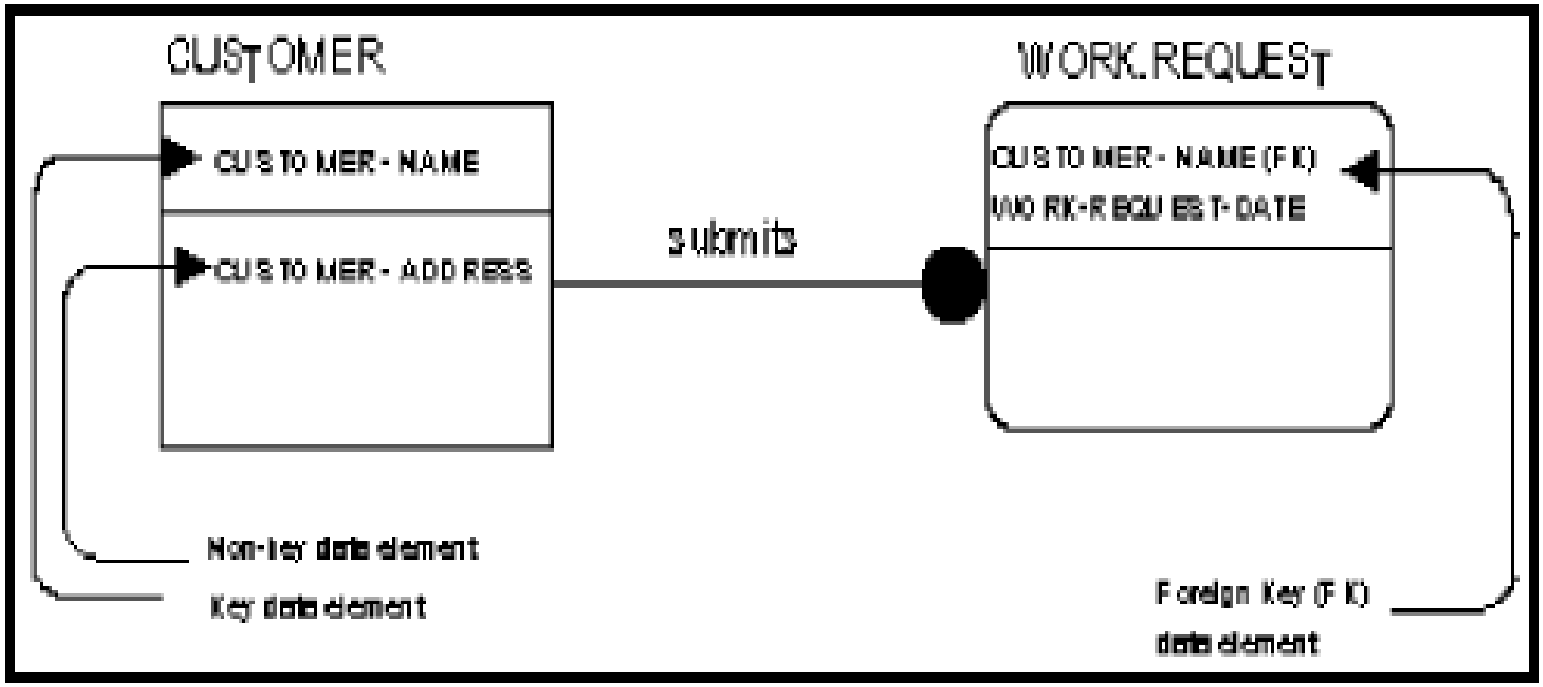


Figure 3-4. Key-Based Model

IDEF1 (information Model) vs. IDEF1x (Data Model)

Information Model	Data Model
focuses on: ◈ Information collected, stored, and managed by the organization ◈ Logical relationships within the organization reflected in the information	focuses on: ◈ Actual data elements in a relational database ◈ Representation & structure of the data
used for: ◈ Problem identification ◈ Requirements definition ◈ Information system design	used for: ◈ Logical design of databases & applications ◈ Physical design of database implementation

PERBEDAAN FOKUS IDEF 1 dan IDEF 1.x

Model Informasi

- Informasi yang dikumpulkan, disimpan, dan dikelola organisasi
- Hubungan logis dalam organisasi tercermin dalam informasi

Model Data

- Elemen data yang sebenarnya dalam basis data relasional
- Representasi & struktur data

PERBEDAAN FUNGSI IDEF 1 dan IDEF 1x

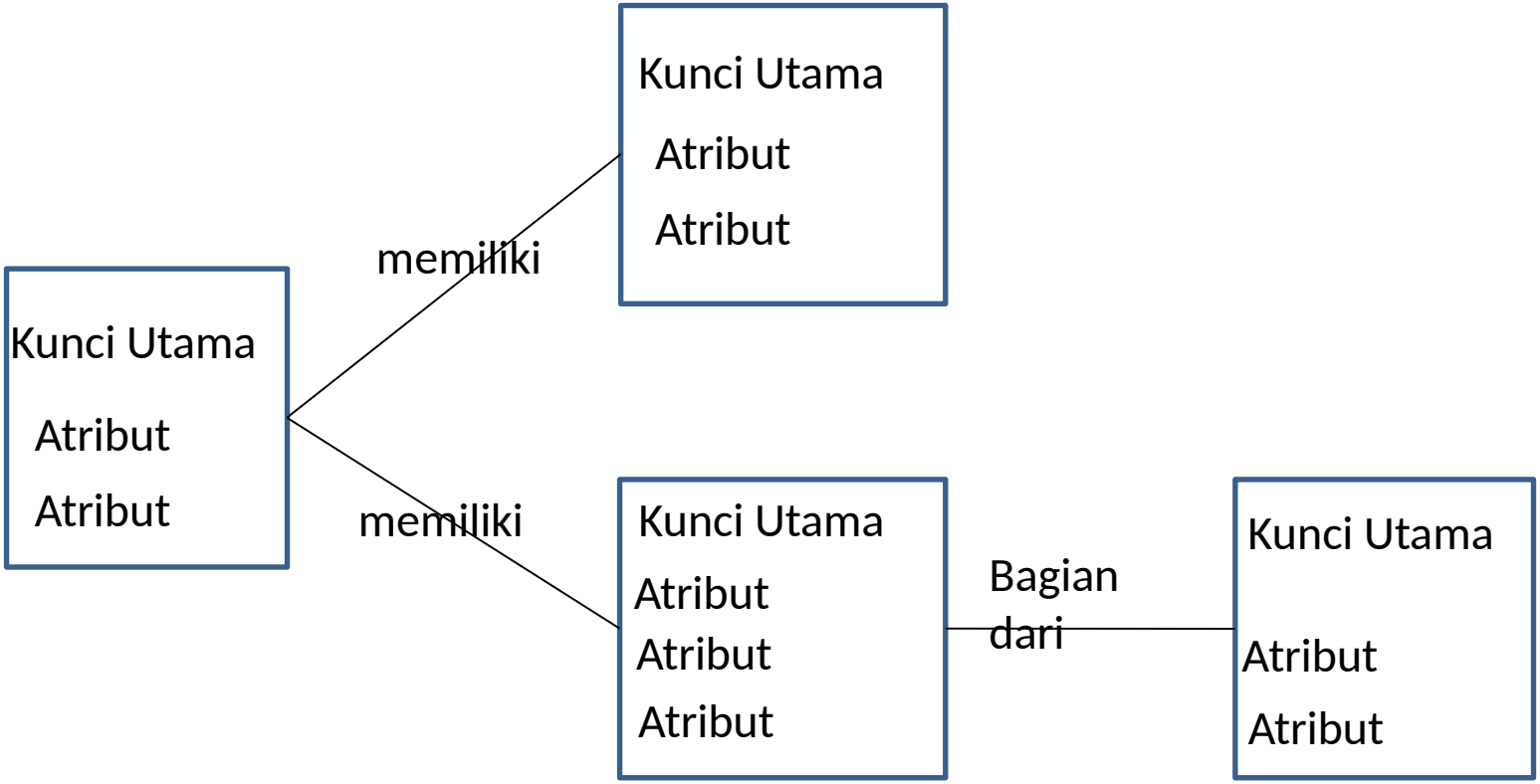
MODEL INFORMASI

- Masalah identifikasi
- Persyaratan definisi Desain sistem informasi

MODEL DATA

- Desain logis dari database & aplikasi
- Desain fisik implementasi database

Langkah pembuatan idef1X

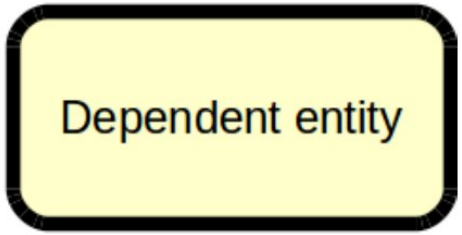


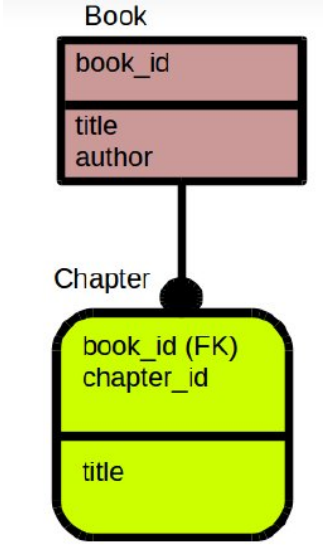
Notasi IDEF1x

- **Entities** .In IDEF1X, entities are either identi er-independent or identi er-dependent. Instances of identi er- independent entities can exist without any other entity instance, while instances of identi er-dependent entities are meaningless (by de nition) without another associated entity instance. Dependence and independence are specific to a model.
- **Connection Relationships**. Connection relationships (solid or dashed lines with lled circles at one or both ends) denote how entities (sets of data instances) relate to one another. The connection relationships are always between exactly two entities. The connection relationship beginning at the independent parent entity and ending at the dependent child entity is labeled with a verb phrase describing the relationship.
- Each connection relationship has an **associated cardinality.** The cardinality specifies the number of instances of the dependent entity that are related to an instance of the independent entity

Jenis ENTITAS

- **Weak entity (dependent)** is represented by a round-cornered rectangle (instances of identifier-dependent entities are meaningless (by definition) without another associated entity instance)
- **Strong entity (independent)** is represented by a rectangle (Instances of identifier-independent entities can exist without any other entity instance)
- Example: book as an independent entity and chapter as a dependent entity. the entity is usually situated above the rectangle.

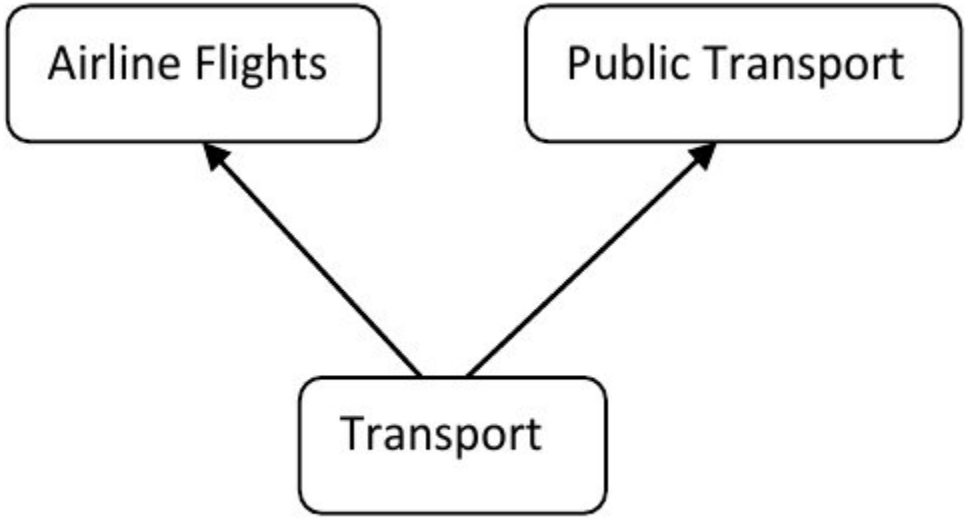




Entitas

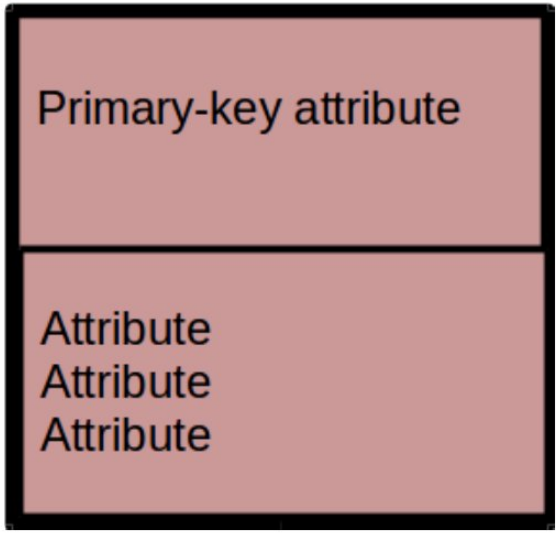
Subject Areas ...

Generic Models ...



Attribute

- Every attribute
  - must have a value (No-Null Rule),
  - and no attribute may have multiple values (No-Repeat Rule).
- Every entity has exactly one primary key displayed above the horizontal line in the entity box.
  - Foreign keys are labeled (FK) and
  - alternative keys are presented as (AK) to show that they are not owned by that entity.



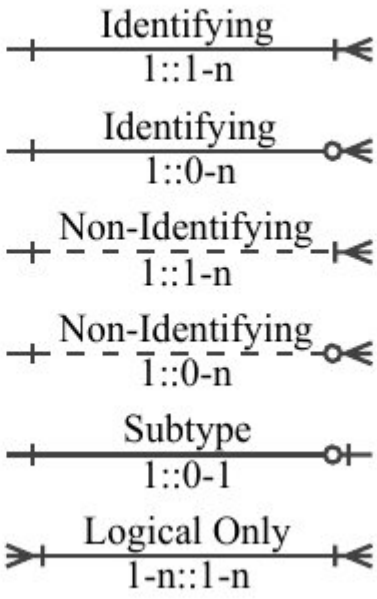
Relationship

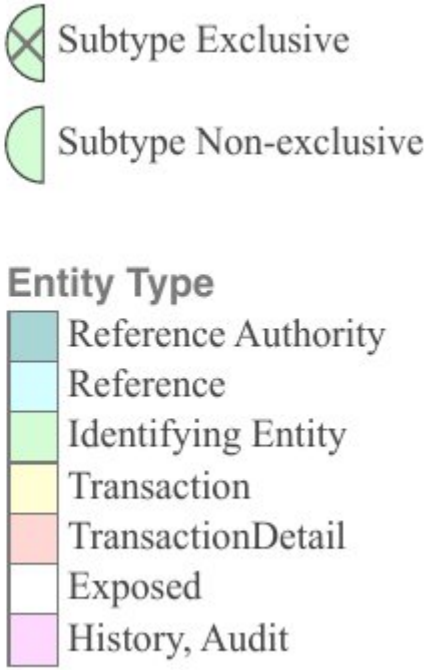
- Categorization Relationships way of distinguishing between the category entities. The category entities are distinguished by a discriminator attribute which must have a different value for each category entity.
- Attributes. Attributes are properties used to describe an entity. Attribute names are unique throughout an IDEF1X model, and the meaning of the names must be consistent. For example, the attribute "color" could have several possible uses for hair color, skin color, or a color in a rainbow. Each use has a range of meaningful values, and thus, the entity must be distinctly named. Each attribute is owned by exactly one entity. The attribute "social security number," for example, could be used in many places in a model, but would be owned by only one entity (e.g., PERSON). Other occurrences of the social security number attributes would be inherited across relations.
- Every attribute must have a value (No-Null Rule), and no attribute may have multiple values (No-Repeat Rule).
- Rules enforce creating proper models. In a situation where it seems that a rule cannot hold, the model is likely wrong.
- Keys. A key is a group of attributes that uniquely identify an entity instance. There are primary and alternate keys. Every entity has exactly one primary key displayed above the horizontal line in the entity box. Entities can have alternate keys that also uniquely identify the entity, but are not used for describing relationships with other entities.
- In a connection relationship, the primary key of the parent migrates to the child. If the relationship is a category relation, the primary key of the child is the same as the generic. If the relationship is an identifying relationship, the primary key of the child must contain attributes inherited from the parent.
- Besides the fact that a key must uniquely identify an entity, all attributes in the key must contribute to the unique identification (Smallest-Key Rule). Thus, when deciding whether an inherited attribute should be part of a key, an issue is whether that attribute is necessary for unique identification. It is not suficient that it contributed to the unique identification of the parent.
- There are also two dependency rules:
  - The Full-Functional-Dependency Rule states that if the primary key iscomposed of multiple attributes, all non-key attributes must be functionally dependent on the entire primarykey.
  - The No-Transitive-Dependency Rule states that every non-key attribute must be functionally dependent only on key attributes.

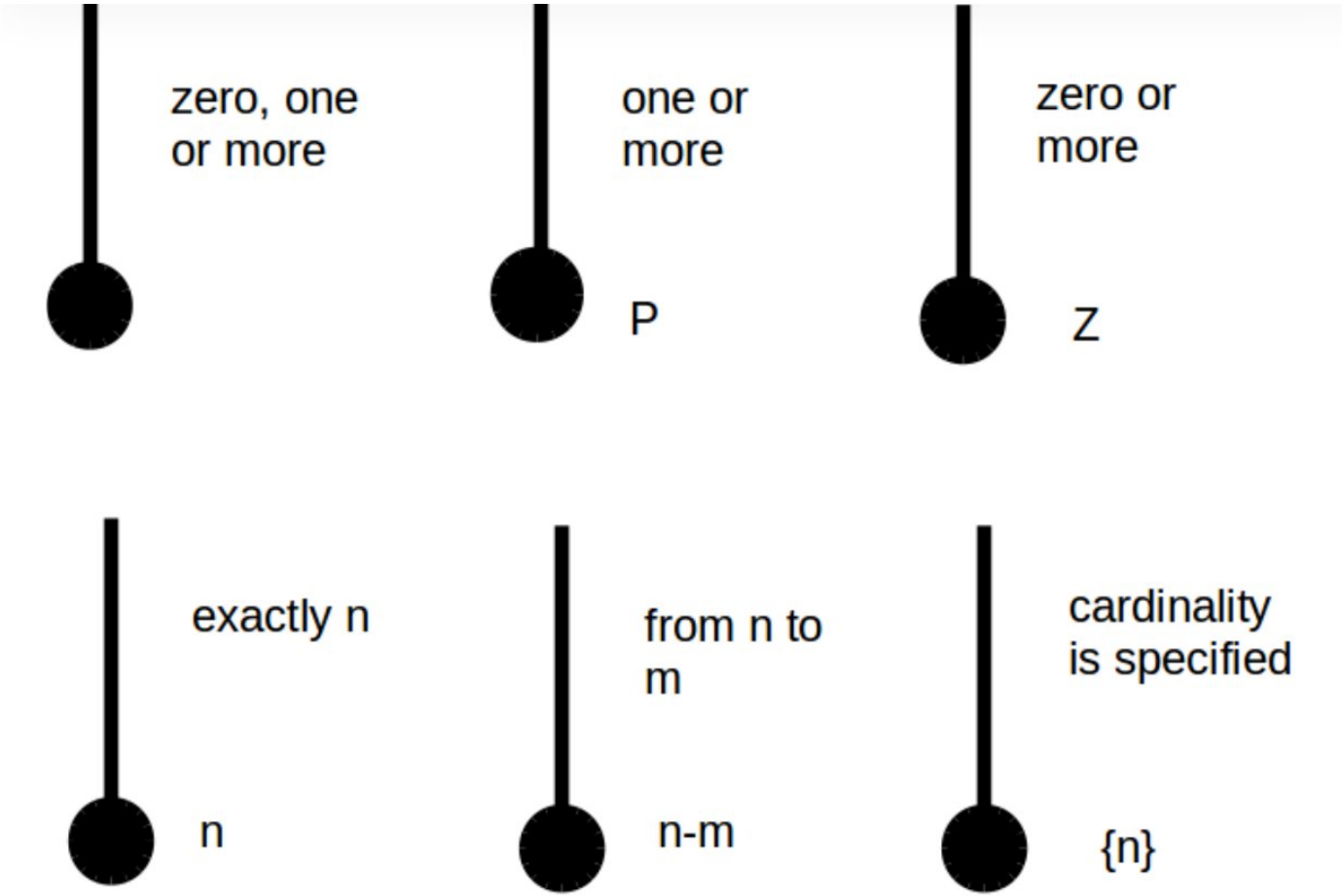
Notasi IDEF 1x

IDEF1X Method & Notation		
Item	**Definition**	**Display**
Key		
Primary Key:	Unique	Above line
Alternate Key:	Other Unique key	AK
Inversion Entry:	Non-Unique index	IE
Entity		
Independent:	Exists without a parent	Square
Dependent:	Dependent on a parent	Round
Relation		
Identifying:	Parent PK forms child PK	Solid line
Non-Identifying:		Broken line
Parent PK is *always* a migrated FK in child		Bold

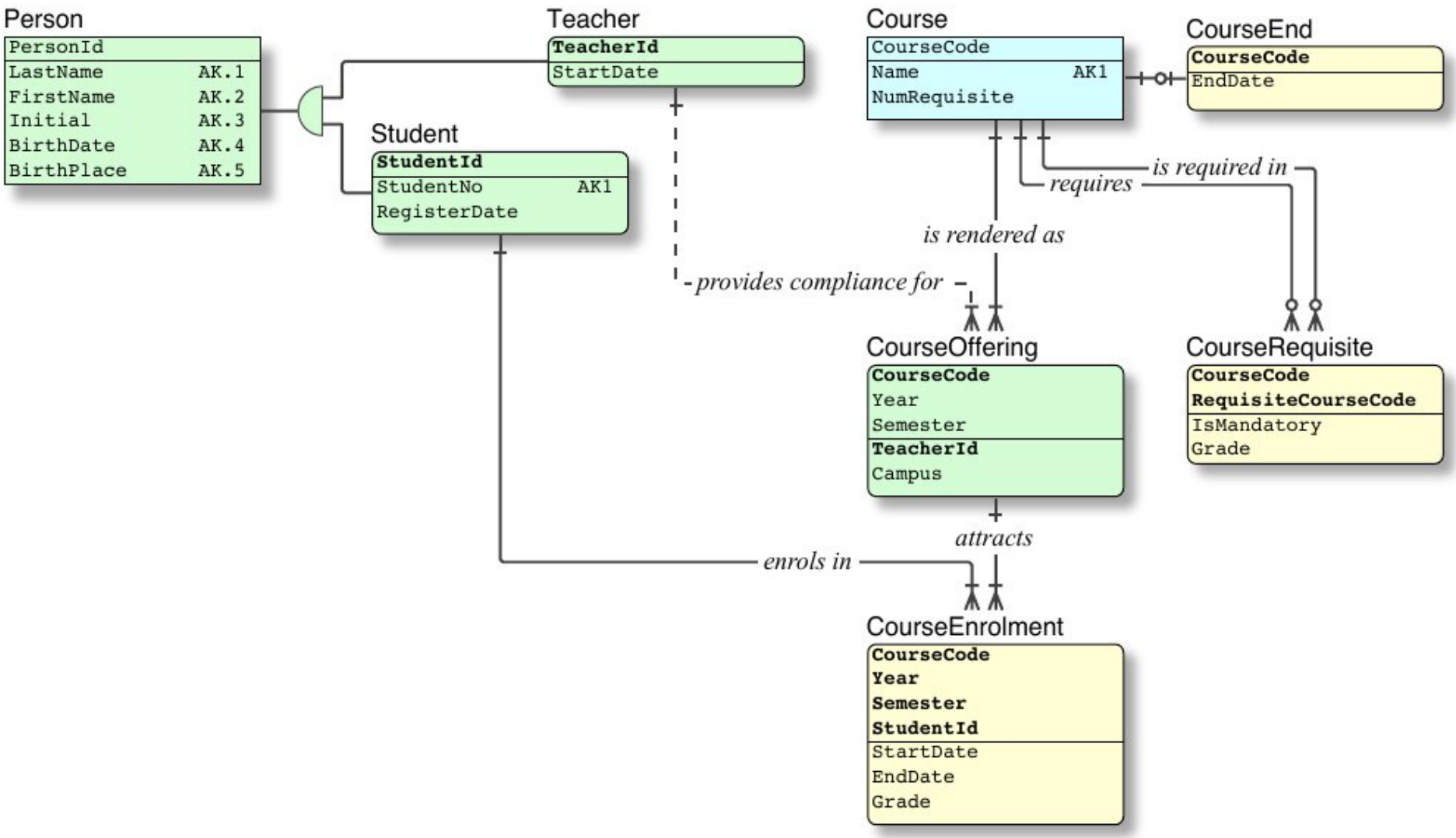
IEEE Relation Notation






zero, one or more
one or more
P
zero or more
Z
exactly n
n
from n to m
n-m
cardinality is specified
{n}

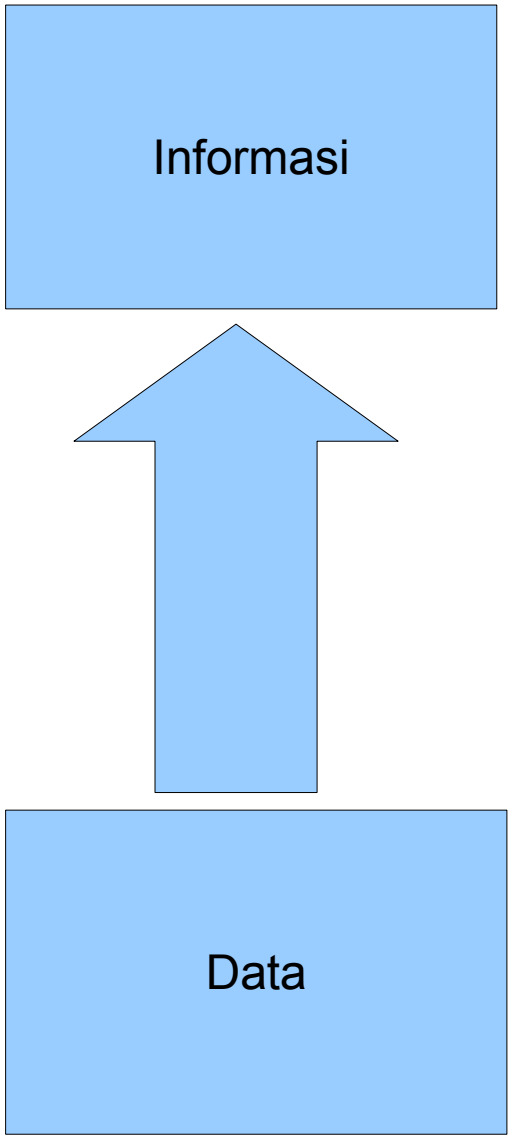
Contoh IDEF 1x



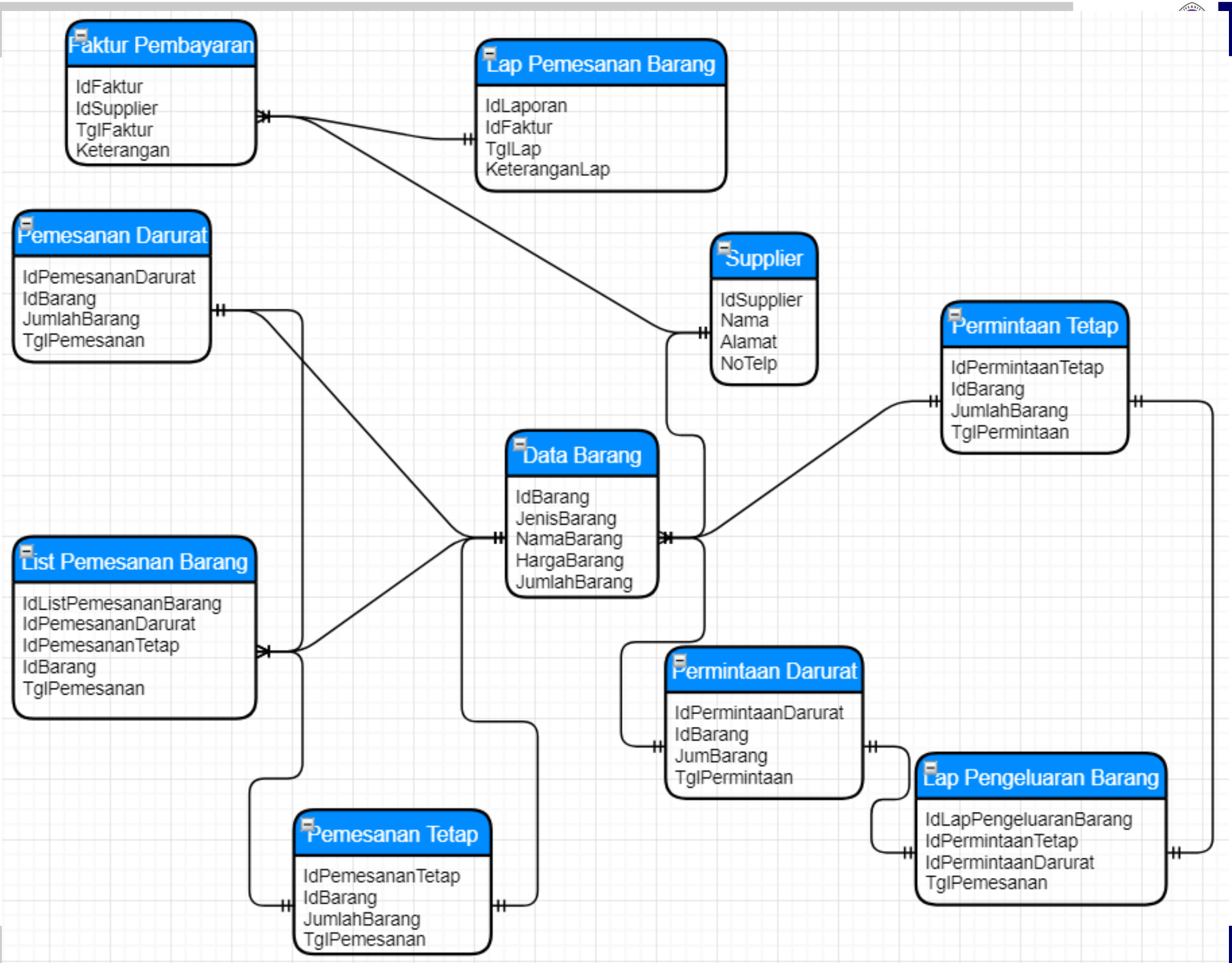
Relationship

- **Relationships**. Solid or dashed lines with filled circles at one or both ends denote how entities relate to one another.
- The relationships are always **between exactly two entities** and are **labeled with a verb phras**e describing the relationship.
- Each connection relationship h**as an associated cardinality** which specifies the number of instances of the dependent entity that are related to an instance of the independent entity.
- This is illustrated in the diagram below:
-

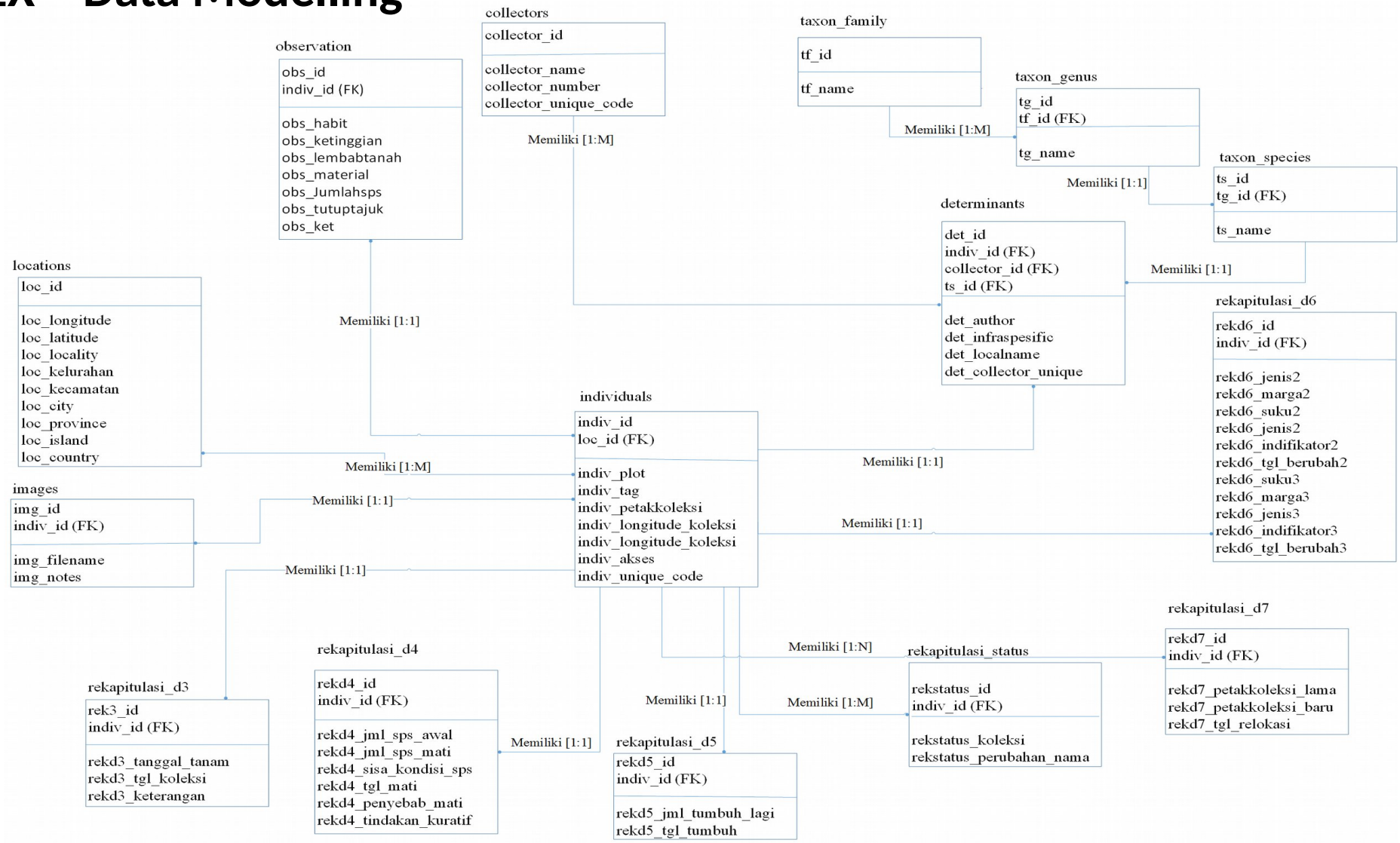
Tahapan menyusun IDEF 1x



- Identifikasi data (tabel) yang menjelaskan entitas pada IDEF1 (berproses untuk menghasilkan entitas IDEF1) → data ini menjadi Entitas pada IDEF1x
- Tentukan atribut (sesuai kolom pada tabel) dari data tersebut, dan tentukan KEY
- Susun matrix relasi dari entitas tersebut
- Tentukan relasi dari tiap Entitas tersebut
- Tentukan kardinalitas dari Entitas

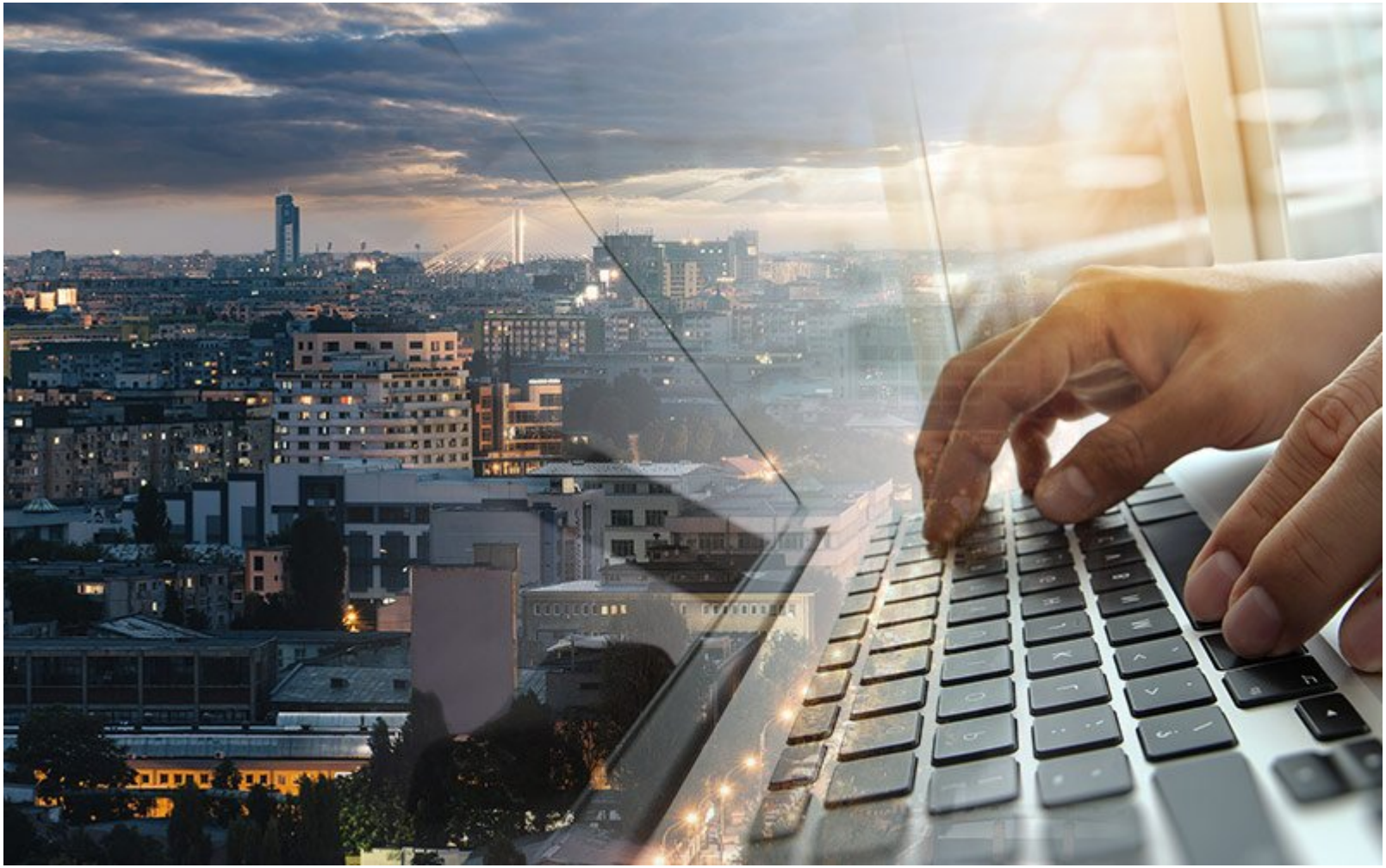

Faktur Pembayaran
IdFaktur
IdSupplier
TglFaktur
Keterangan
Lap Pemesanan Barang
IdLaporan
IdFaktur
TglLap
KeteranganLap
Pemesanan Darurat
IdPemesananDarurat
IdBarang
JumlahBarang
TglPemesanan
Supplier
IdSupplier
Nama
Alamat
NoTelp
Permintaan Tetap
IdPermintaanTetap
IdBarang
JumlahBarang
TglPermintaan
Data Barang
IdBarang
JenisBarang
NamaBarang
HargaBarang
JumlahBarang
List Pemesanan Barang
IdListPemesananBarang
IdPemesananDarurat
IdPemesananTetap
IdBarang
TglPemesanan
Permintaan Darurat
IdPermintaanDarurat
IdBarang
JumBarang
TglPermintaan
Lap Pengeluaran Barang
IdLapPengeluaranBarang
IdPermintaanTetap
IdPermintaanDarurat
TglPemesanan
Pemesanan Tetap
IdPemesananTetap
IdBarang
JumlahBarang
TglPemesanan

IDEF 1X – Data Modelling



Tugas

- Lakukan inventarisasi data-data yang disimpan di dalam layanan organisasi anda
- Petakan data tersebut terkait dengan informasi apa
- Tentukan tabel data tersebut
- Gambarkan relasi tiap tabel



Permodelan Database

SQL

Sejarah singkat SQL standard – ISO/IEC 9075

- 1987 – Initial ISO/IEC Standard
- 1989 – Referential Integrity
- 1992 – SQL2
  - 1995 SQL/CLI (ODBC)
  - 1996 SQL/PSM – Procedural Language extensions
- 1999 – User Defined Types
- 2003 – SQL/XML
- 2008 – Expansions and corrections
- 2011 (or 2012) System Versioned and Application Time Period Tables

- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language, data dimanipulasi dengan semantik SQL, seperti INSERT, DELETE
- Data Definition Language, skema didefinisikan di awal, pendefinisian constraint dengan primary key, foreign key dsb
- Transactions
- Abstraction from physical layer, aplikasi menspesifikasi “apa” bukan “bagaimana”, file bisa berubah tetapi secara logis tidak
- Proprieatry: IBM DB2, Oracle RDMS, Microsoft SQL Server, Sybase SQL Anywhere
- Open Source: MySQL, Postgres, Sqlite

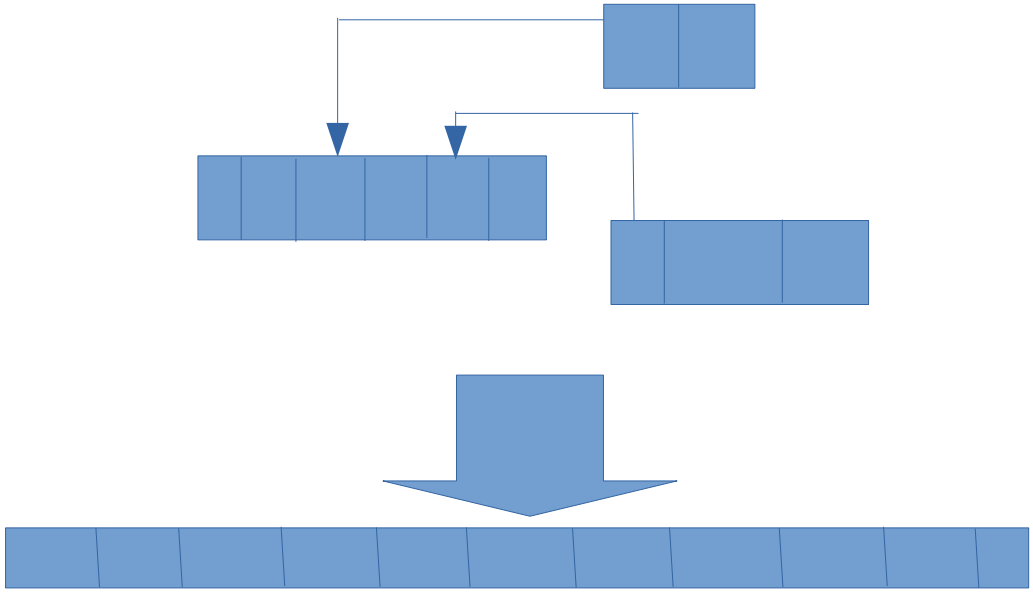
NoSQL Definition

From www.nosql-database.org:

Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontal scalable**.

The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly.

Often more characteristics apply as: **schema-free, easy replication support, simple API, eventually consistent / BASE** (not ACID), a **huge data amount**, and more.



Denormalisasi

- SQL Databases
  - Predefined Schema
  - Standard definition and interface language
  - Tight consistency
  - Well defined semantics

- NoSQL Database
  - No predefined Schema
  - Per-product definition and interface language
  - Getting an answer quickly is more important than getting a correct answer

Kriteria Transaksi Database

ACID

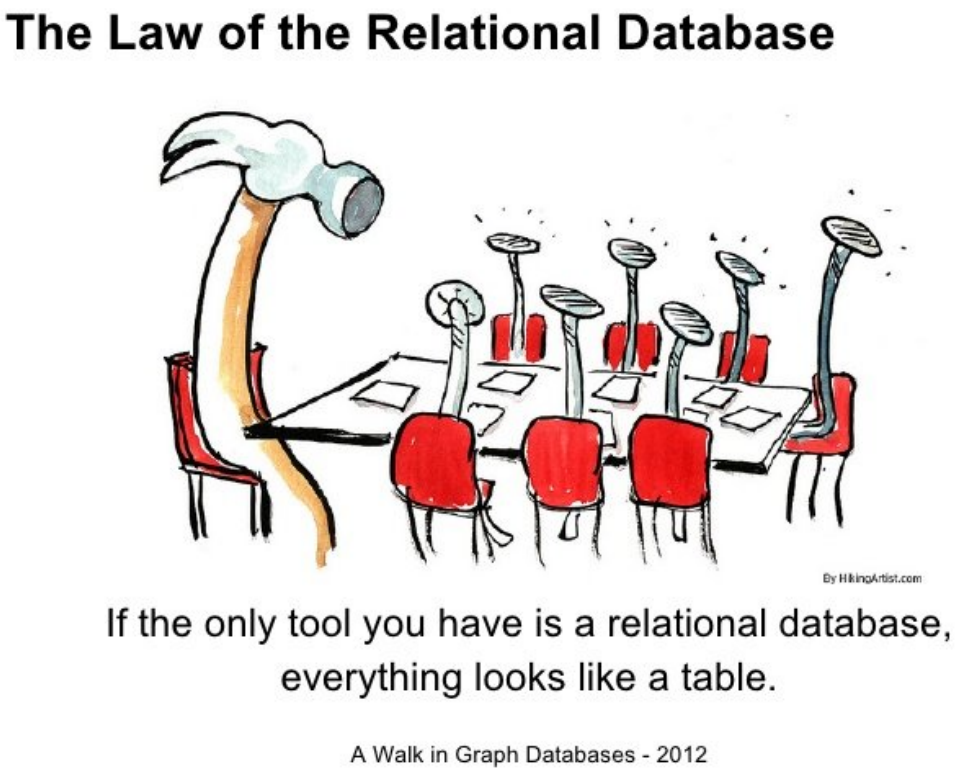
- **Atomic** – Suatu pekerjaan pada sebuah transaksi hanyalah lengkap (commit) atau tidak, ketika pekerjaan itu selesai.
- **Consistent** – Suatu transaksi mentranformasi database dari suatu kondisi konsisten ke kondisi konsisten lainnya. Konsistensi di definisikan dengan constraing *Selalu ?*
- **Isolated** – Hasil dari suatu perubahan yang dilakukan pada suatu transaksi hanya akan terlihat ketika transaksi tersebut telah *commit*
- **Durable** – Hasil dari suatu transaksi yang kommit akan bertahan ketika terjadi gangguan

BASE

- **Basically Available**, satu titik pada sistem terdistribusi bisa down, tetapi seluruh sistem tetap tak berpengaruh
- **Soft state,** (scalable) state dari sistem dari state dan data selalu berubah sepanjang waktu.
- **Eventually Consistent,** dengan waktu tertentu, maka data akan konsisten pada sistem terdistribusi
- Characteristics
  - Weak consistency – stale data OK
  - Availability first
  - Best effort
  - Approximate answers OK
  - Aggressive (optimistic)
  - Simpler and faster

Dorongan menggunakan NoSQL

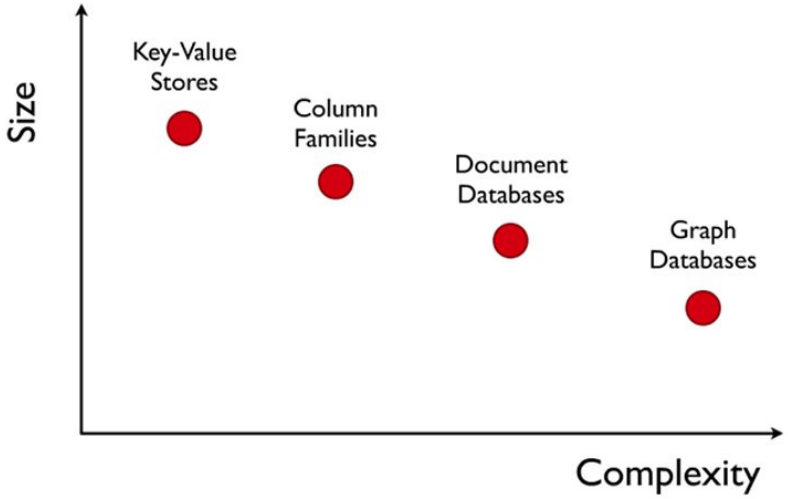
- Masalah RDBMS
  - Tidak ada satupun model yang fit untuk semua masalah
  - Mismatch dengan model obyek
  - Sulit terskala, replikasi dan joint antar node sangat sulit
  - Disain skema data yang rigid
  - Tapi transaction dan ACID mempermudah pekerjaan, serta banyak orang sudah familiar
- Dorongan pemilihan NoSQL
  - Big Data
  - Scalability
  - Data format
  - Manageabilityi

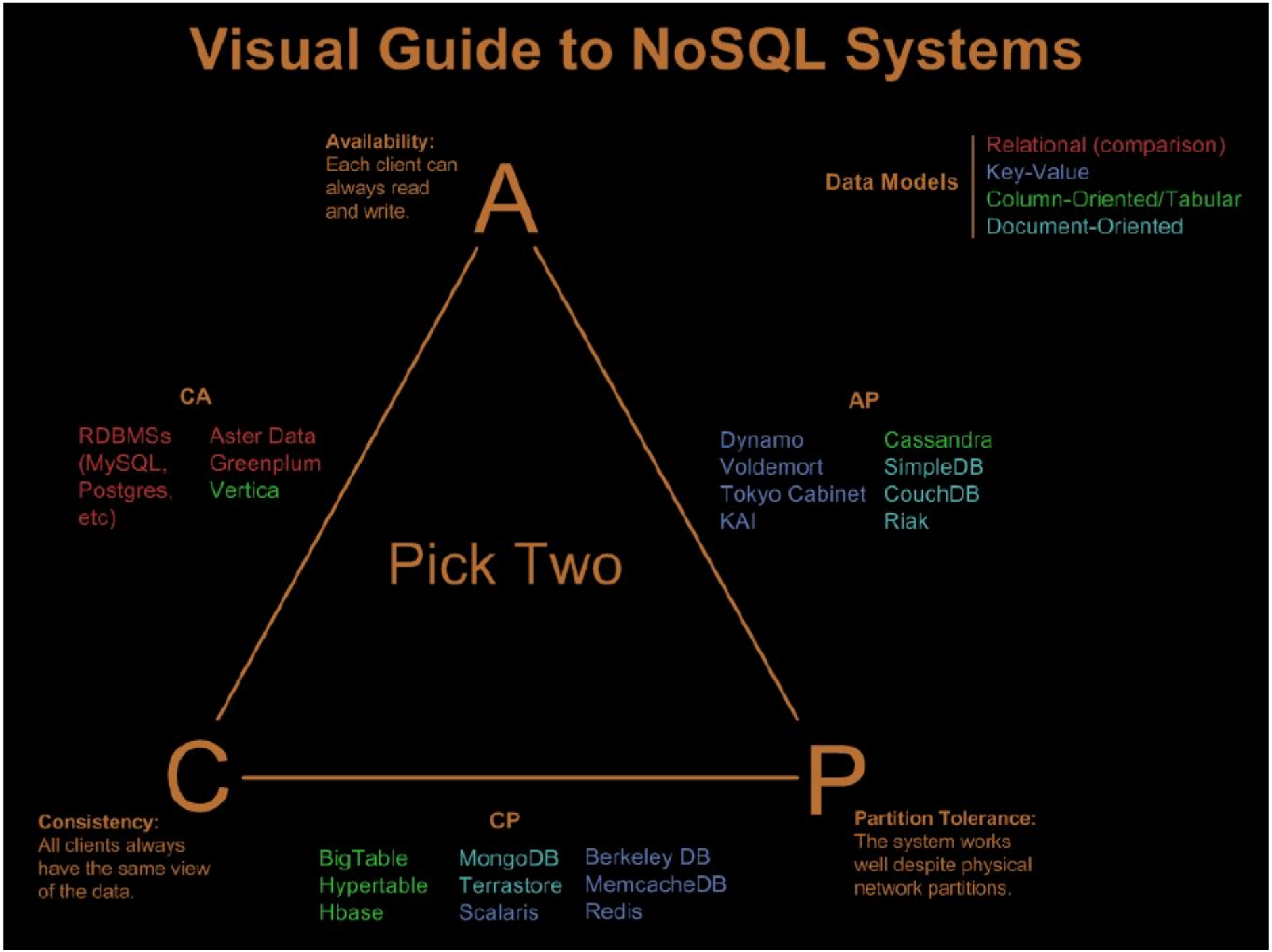


NoSQL Distinguishing Characteristics

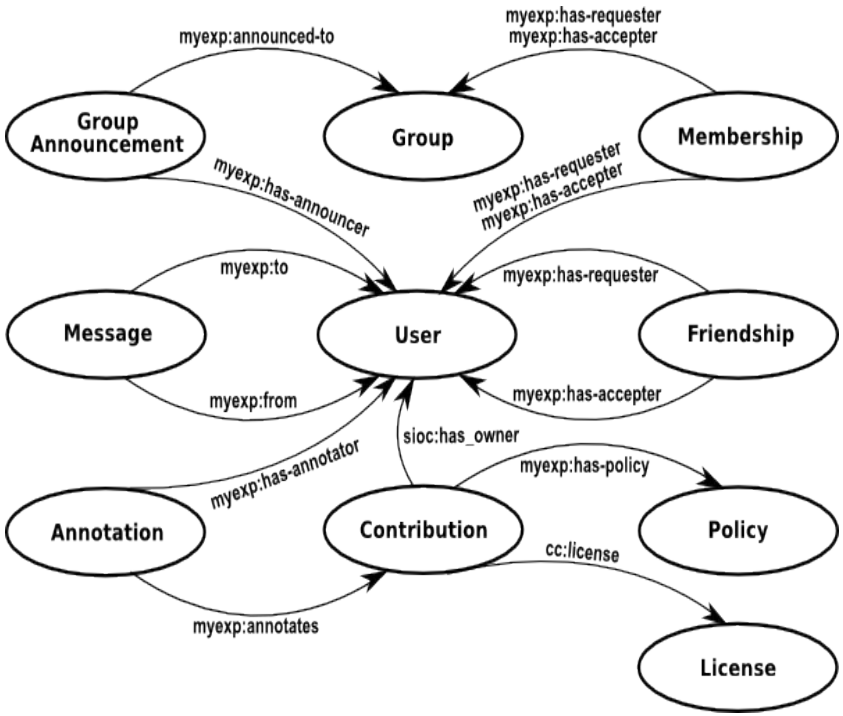
- NoSQL databases reject:
  - Overhead of ACID transactions
  - "Complexity" of SQL
  - Burden of up-front schema design
  - Declarative query expression
  - Yesterday's technology
- Programmer responsible for
  - Step-by-step procedural language
  - Navigating access path

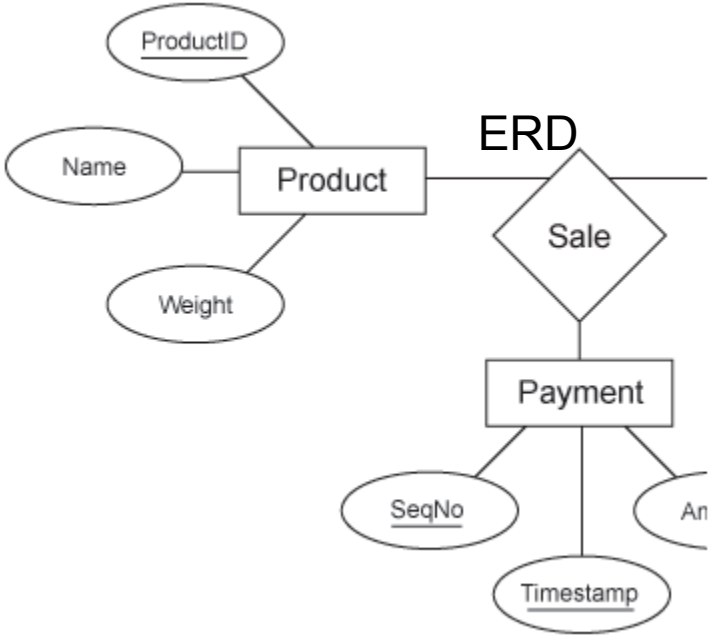
- Key/Value Pairs: Memcached, Riak, Redis, Voldemort
- Tuples (rows).
- Documents: MongoDB, CouchDB
- Columns: Cassandra
- Objects:Objectivity, DB4O
- Graphs: Neo4J,

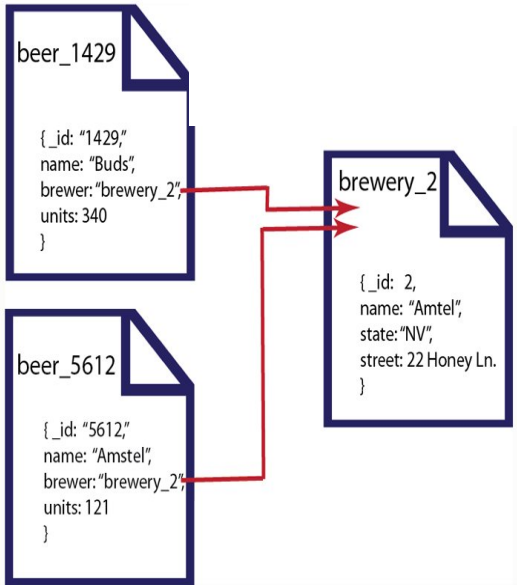



Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

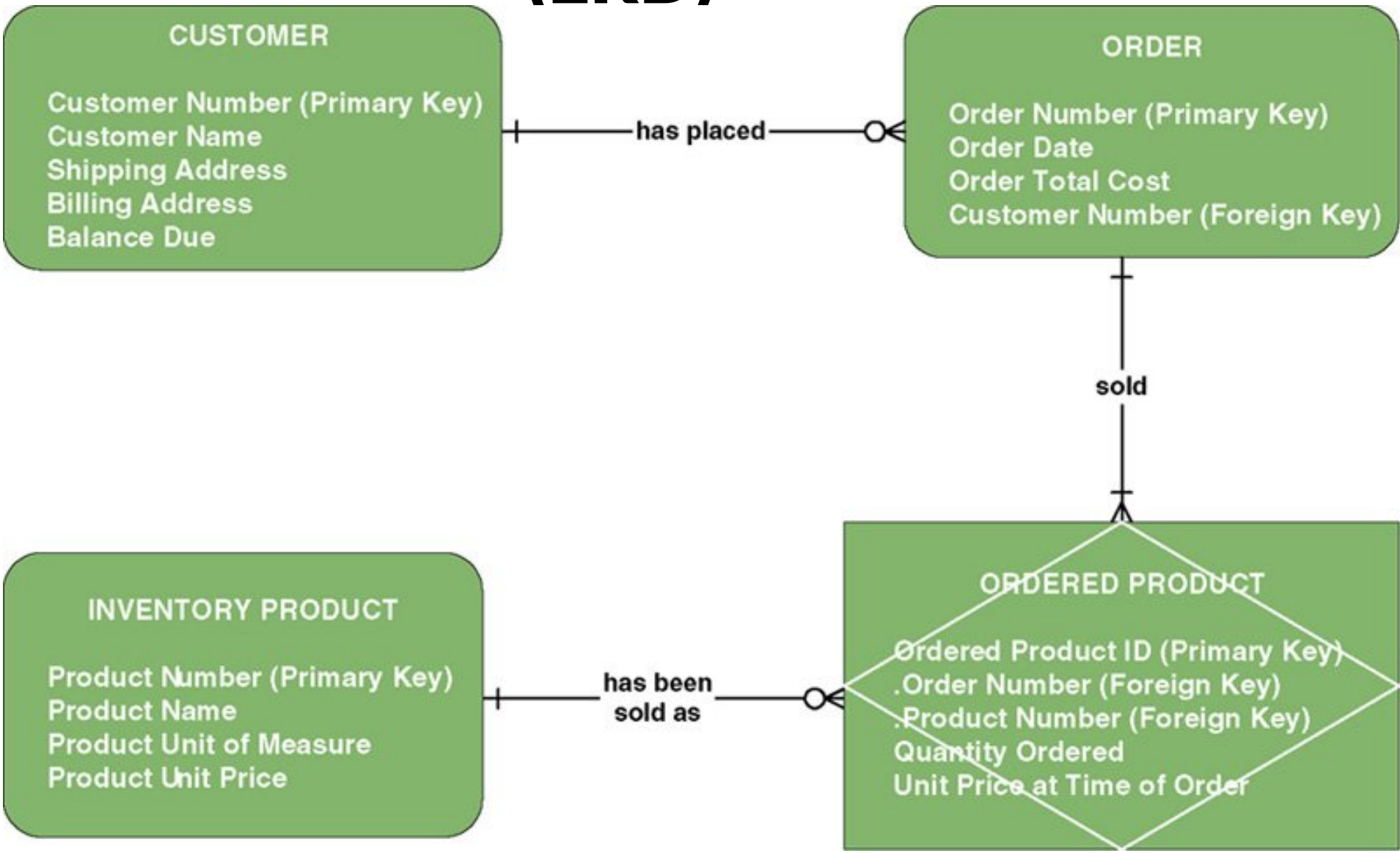
Notasi yang tepat







Sample Entity Relationship Diagram (ERD)



Data Modeling Concepts: Entity

Entity – a class of persons, places, objects, events, or concepts about which we need to capture and store data.

- Named by a singular noun

  - Persons: agency, contractor, customer, department, division, employee, instructor, student, supplier.
  - Places: sales region, building, room, branch office, campus.
  - Objects: book, machine, part, product, raw material, software license, software package, tool, vehicle model, vehicle.
  - Events: application, award, cancellation, class, flight, invoice, order, registration, renewal, requisition, reservation, sale, trip.
  - Concepts: account, block of time, bond, course, fund, qualification, stock.

Data Modeling Concepts: Entity

Entity instance – a single occurrence of an entity.

entity

instances

Student ID	Last Name	First Name
2144	Arnold	Betty
3122	Taylor	John
3843	Simmons	Lisa
9844	Macy	Bill
2837	Leath	Heather
2293	Wrench	Tim

Entities

- An **entity** is something about which we want to store data.
- An **entity** is a class of persons, places, objects, events, or concepts about which we need to capture and store data.
- An **entity instance** is a single occurrence of an entity.

Entity Discovery

- In interviews or JRP sessions, pay attention to key words (i.e. "we need to keep track of ...").
- In interviews or JRP sessions, ask users to identify things about which they would like to capture, store, and produce information.
- Study existing forms, files, and reports.
- Scan use case narratives for nouns.
- Some CASE tools can reverse engineer existing files and databases.

Data Modeling Concepts: Attributes

Attribute – a descriptive property or characteristic of an entity. Synonyms include *element*, *property*, and *field*.

- Just as a physical student can have attributes, such as hair color, height, etc., data entity has data attributes

Compound attribute – an attribute that consists of other attributes. Synonyms in different data modeling languages are numerous: concatenated attribute, composite attribute, and data structure.

Attributes

- An **attribute** is a descriptive property or characteristic of an entity. Synonyms include *element*, *property*, and *field*

Attributes - Identification

- Every entity must have an identifier or key.
  - An **key** is an attribute, or a group of attributes, which assumes a unique value for each entity instance. It is sometimes called an *identifier*.
- Sometimes more than one attribute is required to uniquely identify an instance of an entity.
  - A group of attributes that uniquely identifies an instance of an entity is called a **concatenated key**. Synonyms include *composite key* and *compound key*.

Data Modeling Concepts: Data Type

Data type – a property of an attribute that identifies what type of data can be stored in that attribute.

Representative Logical Data Types for Attributes	
Data Type	**Logical Business Meaning**
NUMBER	Any number, real or integer.
TEXT	A string of characters, inclusive of numbers. When numbers are included in a TEXT attribute, it means that we do not expect to perform arithmetic or comparisons with those numbers.
MEMO	Same as TEXT but of an indeterminate size. Some business systems require the ability to attach potentially lengthy notes to a give database record.
DATE	Any date in any format.
TIME	Any time in any format.
YES/NO	An attribute that can assume only one of these two values.
VALUE SET	A finite set of values. In most cases, a coding scheme would be established (e.g., FR=Freshman, SO=Sophomore, JR=Junior, SR=Senior).
IMAGE	Any picture or image.

Data Modeling Concepts: Domains

Domain – a property of an attribute that defines what values an attribute can legitimately take on.

Representative Logical Domains for Logical Data Types		
Data Type	**Domain**	**Examples**
NUMBER	For integers, specify the range. For real numbers, specify the range and precision.	{10-99} {1.000-799.999}
TEXT	Maximum size of attribute. Actual values usually infinite; however, users may specify certain narrative restrictions.	Text(30)
DATE	Variation on the MMDDYYYY format.	MMDDYYYY MMYYYY
TIME	For AM/PM times: HHMMT For military (24-hour times): HHMM	HHMMT HHMM
YES/NO	{YES, NO}	{YES, NO} {ON, OFF}
VALUE SET	{value#1, value#2,…value#n} {table of codes and meanings}	{M=Male F=Female}

Data Modeling Concepts: Default Value

Default value – the value that will be recorded if a value is not specified by the user.

Permissible Default Values for Attributes		
Default Value	**Interpretation**	**Examples**
A legal value from the domain	For an instance of the attribute, if the user does not specify a value, then use this value.	0 1.00
NONE or NULL	For an instance of the attribute, if the user does not specify a value, then leave it blank.	NONE NULL
Required or NOT NULL	For an instance of the attribute, require that the user enter a legal value from the domain. (This is used when no value in the domain is common enough to be a default but some value must be entered.)	REQUIRED NOT NULL

Data Modeling Concepts: Identification

Key – an attribute, or a group of attributes, that assumes a unique value for each entity instance. It is sometimes called an *identifier*.

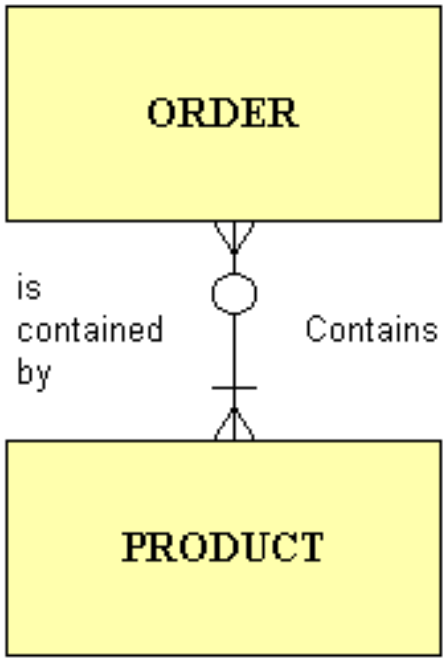
- **Concatenated key** - group of attributes that uniquely identifies an instance. Synonyms: composite key, compound key.
- **Candidate key** – one of a number of keys that may serve as the primary key. Synonym: *candidate identifier*.
- **Primary key** – a candidate key used to uniquely identify a single entity instance.
- **Alternate key** – a candidate key not selected to become the primary key. Synonym: secondary key.

Relationships

- A **relationship** is a natural business association that exists between one or more entities. The relationship may represent an event that links the entities, or merely a logical affinity that exists between the entities.
  - A connecting line between two entities on an ERD represents a relationship.
  - A verb phrase describes the relationship.
    - All relationships are implicitly bidirectional, meaning that they can interpreted in both directions.

Visible Analyst - [Ordered Product:ERD]

File Edit View Options Repository Diagram Tools Window Help



100% PIA-CURRENT

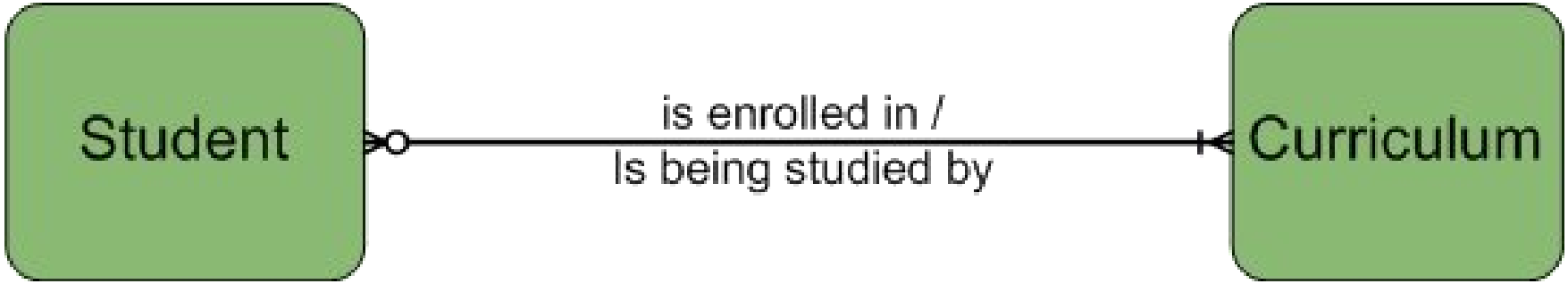
Data Modeling Concepts: Subsetting Criteria

Subsetting criteria – an attribute(s) whose finite values divide all entity instances into useful subsets. Sometimes called an inversion entry.

Data Modeling Concepts: Relationships

Relationship – a natural business association that exists between one or more entities.

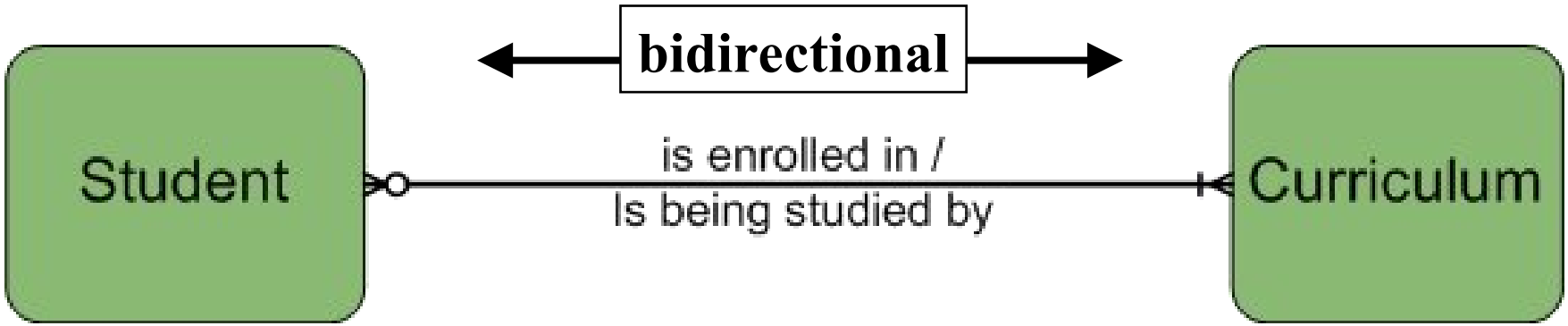
The relationship may represent an event that links the entities or merely a logical affinity that exists between the entities.



Data Modeling Concepts: Cardinality

Cardinality – the minimum and maximum number of occurrences of one entity that may be related to a single occurrence of the other entity.

Because all relationships are bidirectional, cardinality must be defined in both directions for every relationship.



Cardinality

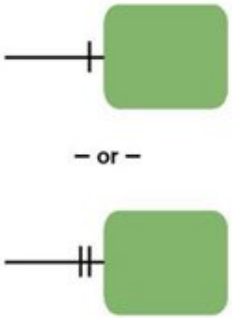
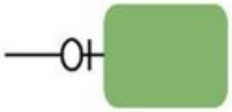
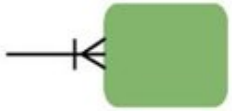
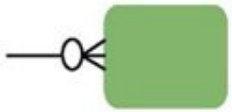
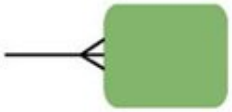
- Each relationship on an ERD also depicts the complexity or degree of each relationship, and this is called ***cardinality***.
  - **Cardinality** defines the minimum and maximum number of occurrences of one entity for a single occurrence of the related entity. Because all relationships are bi-directional, cardinality must be defined in both directions for every relationship.

Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one	1	1	
Zero or one	0	1	
One or more	1	many (> 1)	
Zero, one, or more	0	many (> 1)	
More than one	> 1	> 1	

Figure 5.3

Cardinality Notations

Cardinality Notations

Cardinality Interpretation	Minimum Instances	Maximum Instances	Graphic Notation
Exactly one (one and only one)	1	1	
Zero or one	0	1	
One or more	1	many (>1)	
Zero, one, or more	0	many (>1)	
More than one	>1	>1	

Data Modeling Concepts: Degree

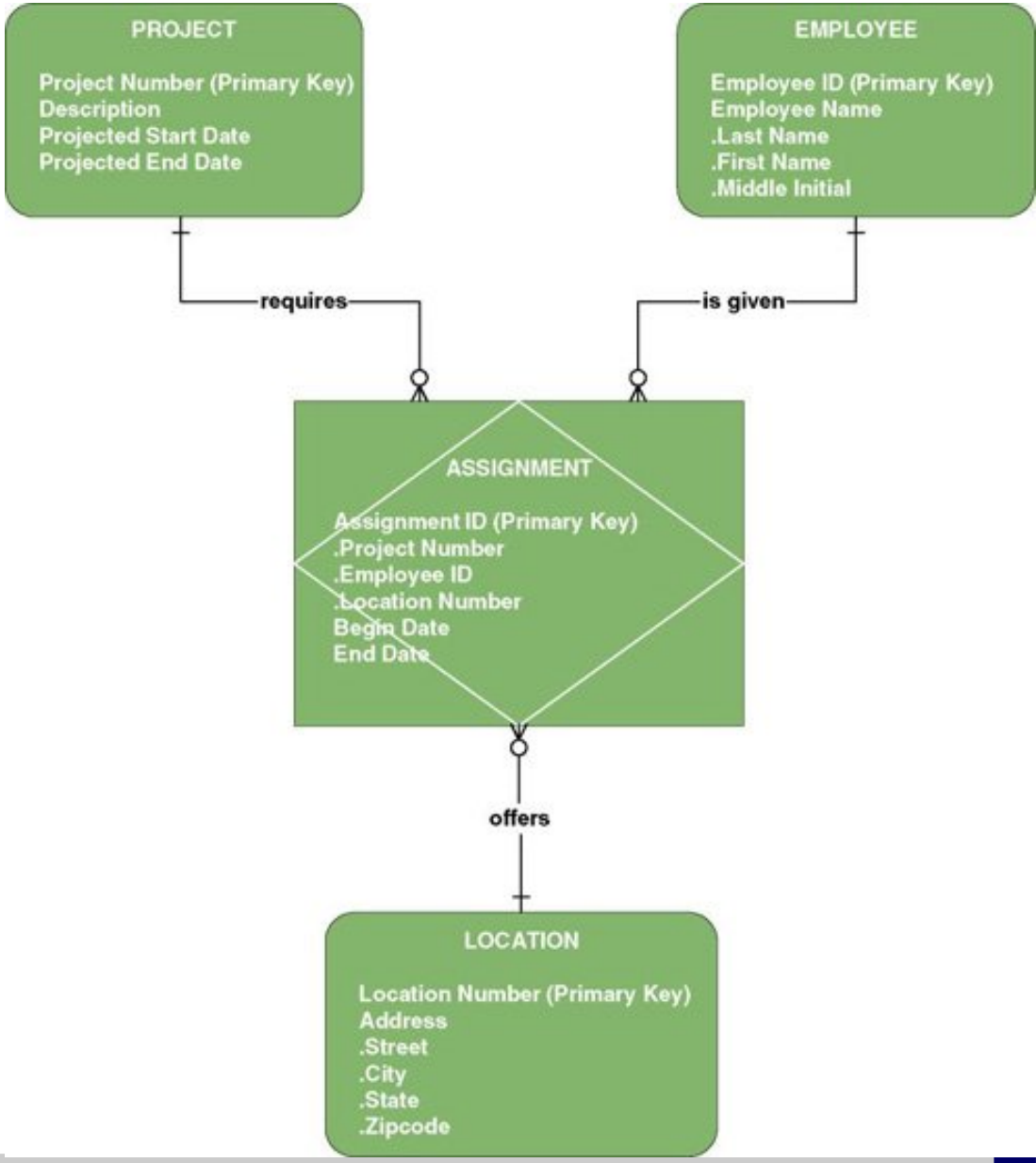
Degree – the number of entities that participate in the relationship.

- A relationship between two entities is called a *binary relationship*.
- A relationship between three entities is called a *3-ary* or *ternary relationship*.
- A relationship between different instances of the same entity is called a *recursive relationship*.

Data Modeling Concepts: Degree

Relationships may exist between more than two entities and are called *N-ary* relationships.

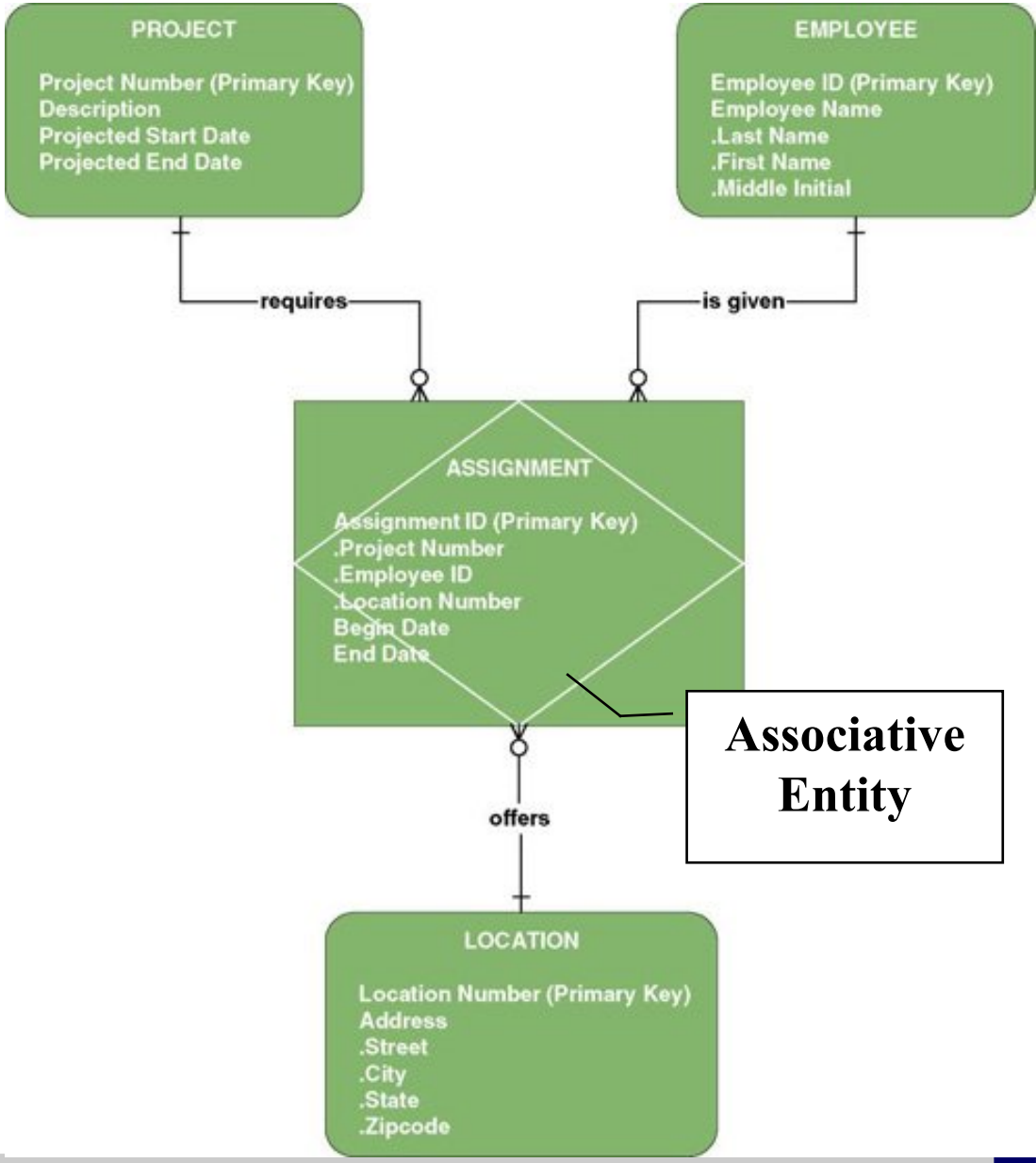
The example ERD depicts a *ternary relationship*.



Data Modeling Concepts: Degree

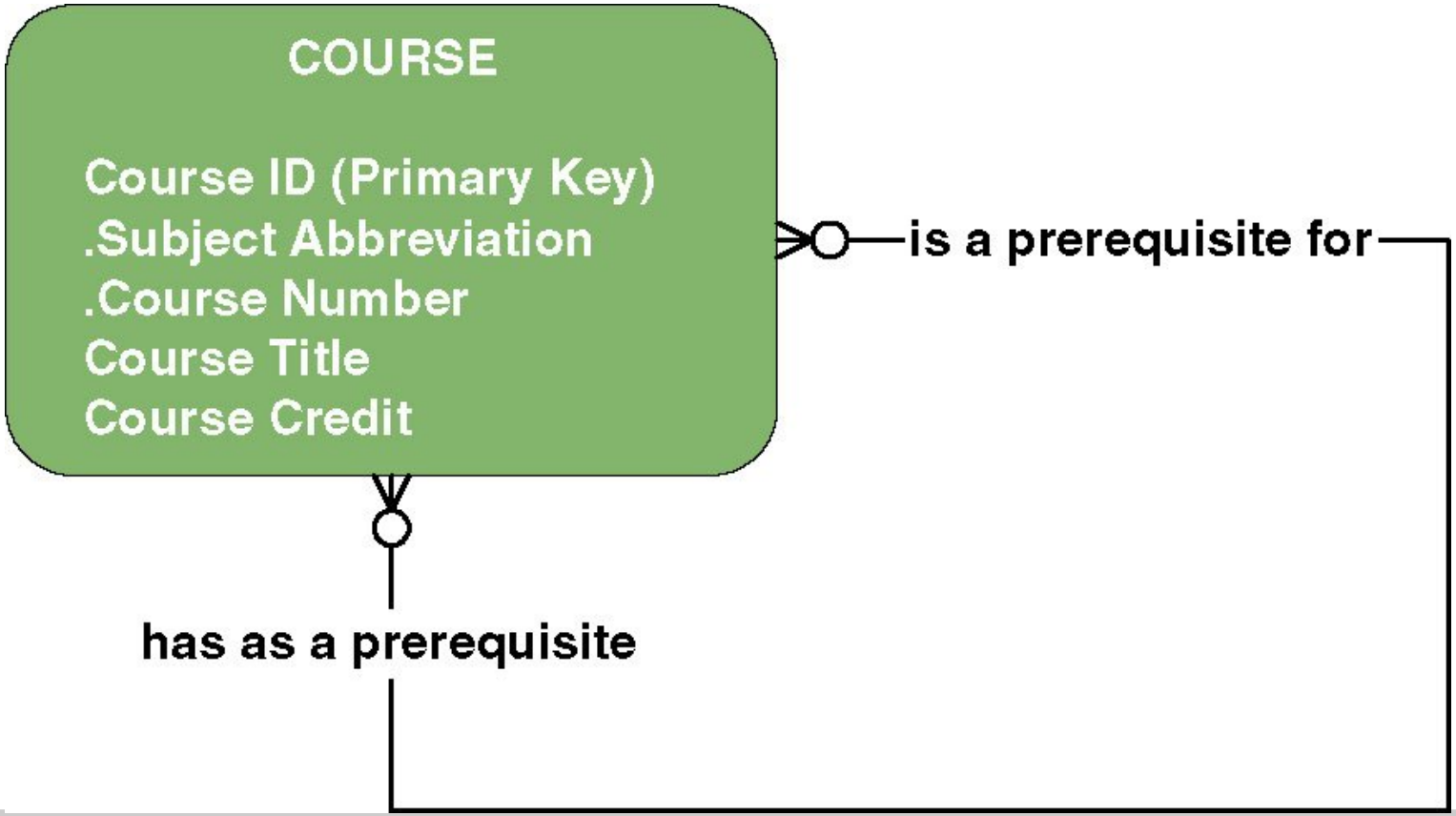
Associative entity – an entity that inherits its primary key from more than one other entity (called parents).

Each part of that concatenated key points to one and only one instance of each of the connecting entities.



Data Modeling Concepts: Recursive Relationship

Recursive relationship - a relationship that exists between instances of the same entity



Data Modeling Concepts: Foreign Keys

Foreign key – a primary key of an entity that is used in another entity to identify instances of a relationship.

- A foreign key is a primary key of one entity that is contributed to (duplicated in) another entity to identify instances of a relationship.
- A foreign key always matches the primary key in the another entity
- A foreign key may or may not be unique (generally not)
- The entity with the foreign key is called the child.
- The entity with the matching primary key is called the parent.

Data Modeling Concepts: Parent and Child Entities

Parent entity - a data entity that contributes one or more attributes to another entity, called the child. In a one-to-many relationship the parent is the entity on the "one" side.

Child entity - a data entity that derives one or more attributes from another entity, called the parent. In a one-to-many relationship the child is the entity on the "many" side.

Data Modeling Concepts: Foreign Keys

Primary Key

Student ID	Last Name	First Name	Dorm
2144	Arnold	Betty	Smith
3122	Taylor	John	Jones
3843	Simmons	Lisa	Smith
9844	Macy	Bill	
2837	Leath	Heather	Smith
2293	Wrench	Tim	Jones

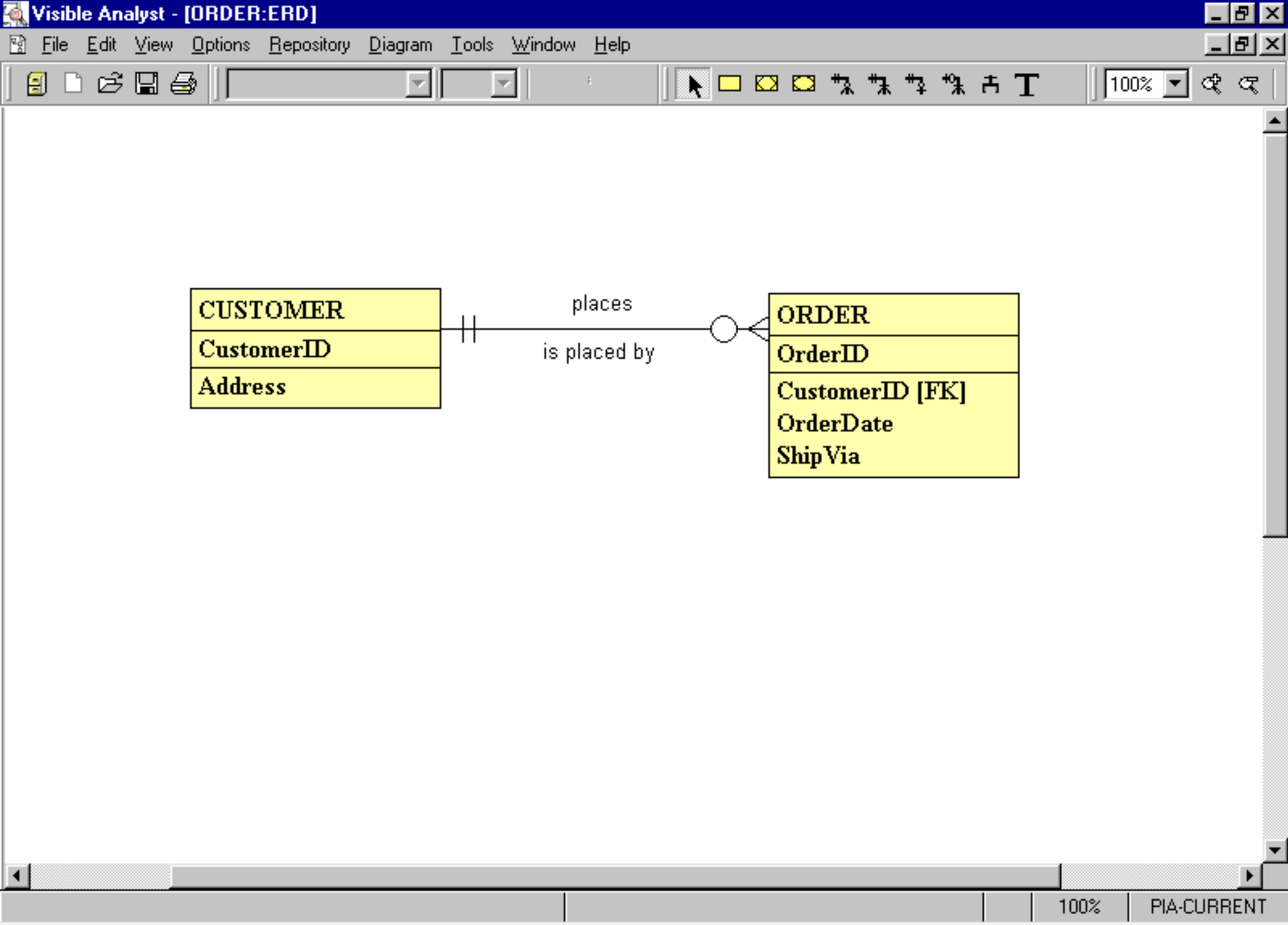
Primary Key

Dorm	Residence Director
Smith	Andrea Fernandez
Jones	Daniel Abidjan

Foreign Key Duplicated from primary key of Dorm entity (not unique in Student entity)

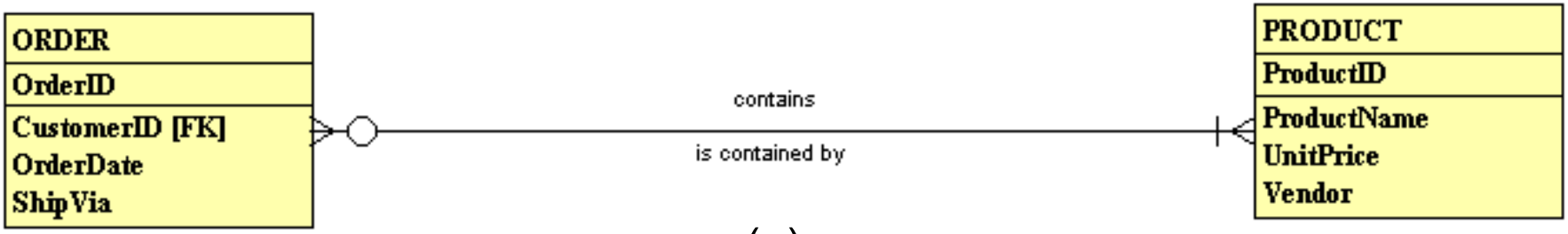
Foreign Keys

- A relationship implies that instances of one entity are related to instances of another entity.
- To be able to identify those instances for any given entity, the primary key of one entity must be migrated into the other entity as a ***foreign key***.
  - A **foreign key** is a primary key of one entity that is contributed to (duplicated in) another entity for the purpose of identifying instances of a relationship. A foreign key (always in a child entity) always matches the primary key (in a parent entity).

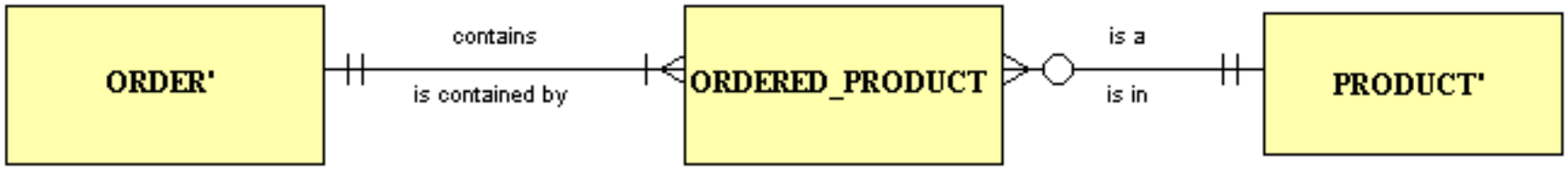

Visible Analyst - [ORDER:ERD]
File Edit View Options Repository Diagram Tools Window Help
100%
CUSTOMER
CustomerID
Address
places
is placed by
ORDER
OrderID
CustomerID [FK]
OrderDate
ShipVia
100%
PIA-CURRENT

Visible Analyst - [Ordered Product:ERD]

File Edit View Options Repository Diagram Tools Window Help



(a)



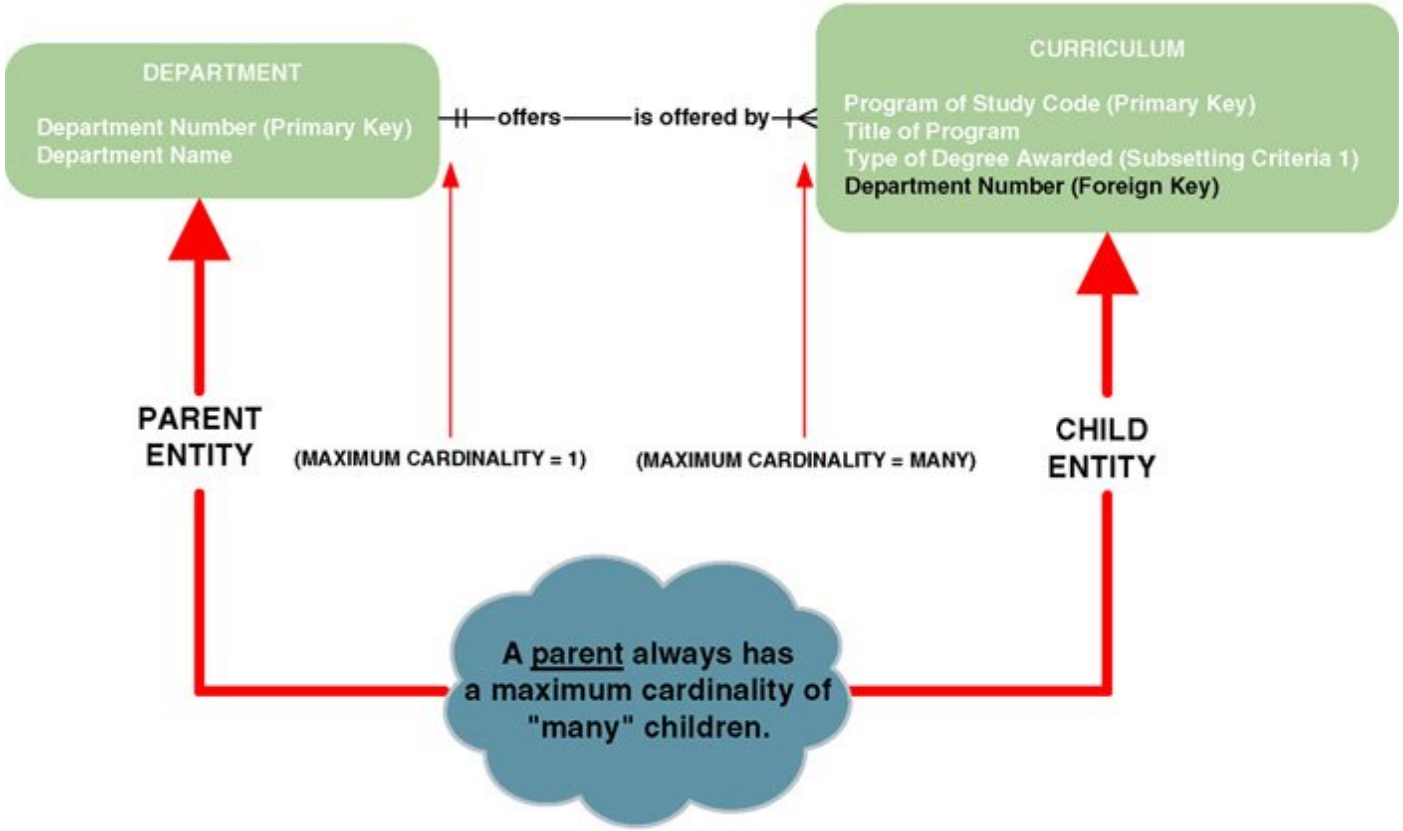
(b)

80% PIA-CURRENT

Data Modeling Concepts: Nonidentifying Relationships

Nonidentifying relationship – relationship where each participating entity has its own independent primary key

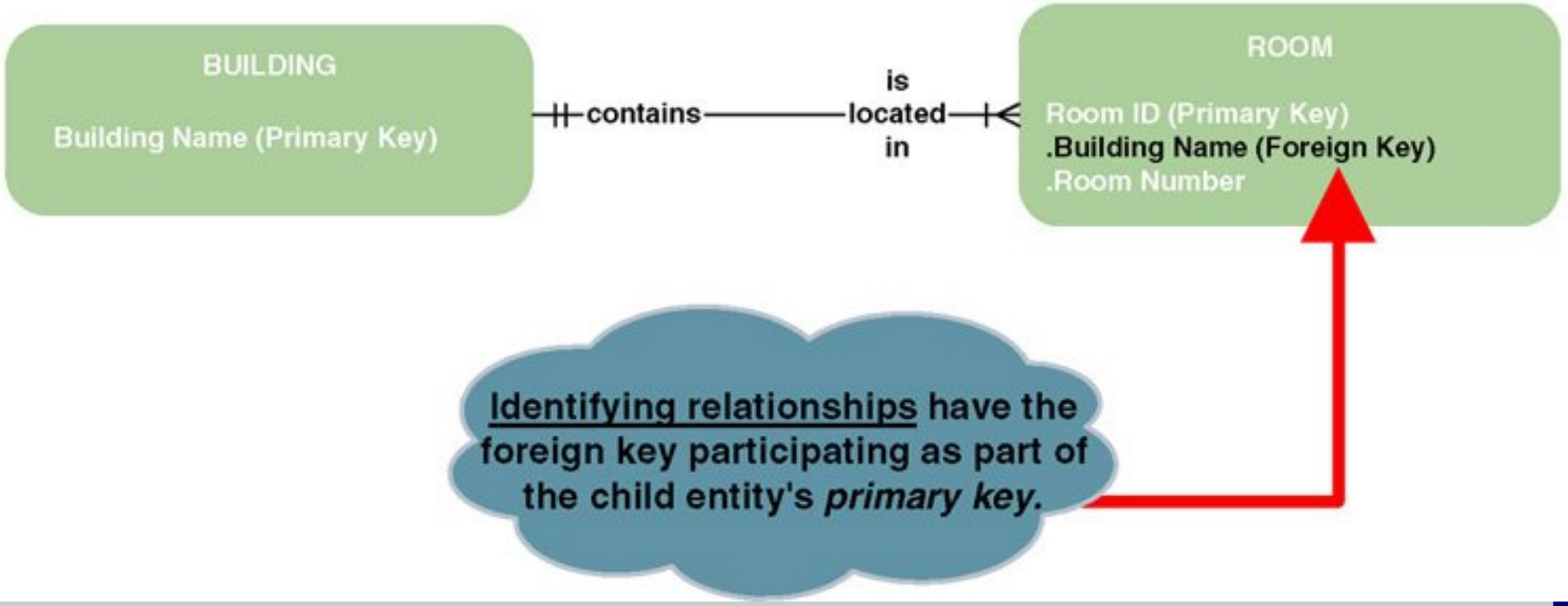
- Primary key attributes are not shared.
- The entities are called *strong* entities



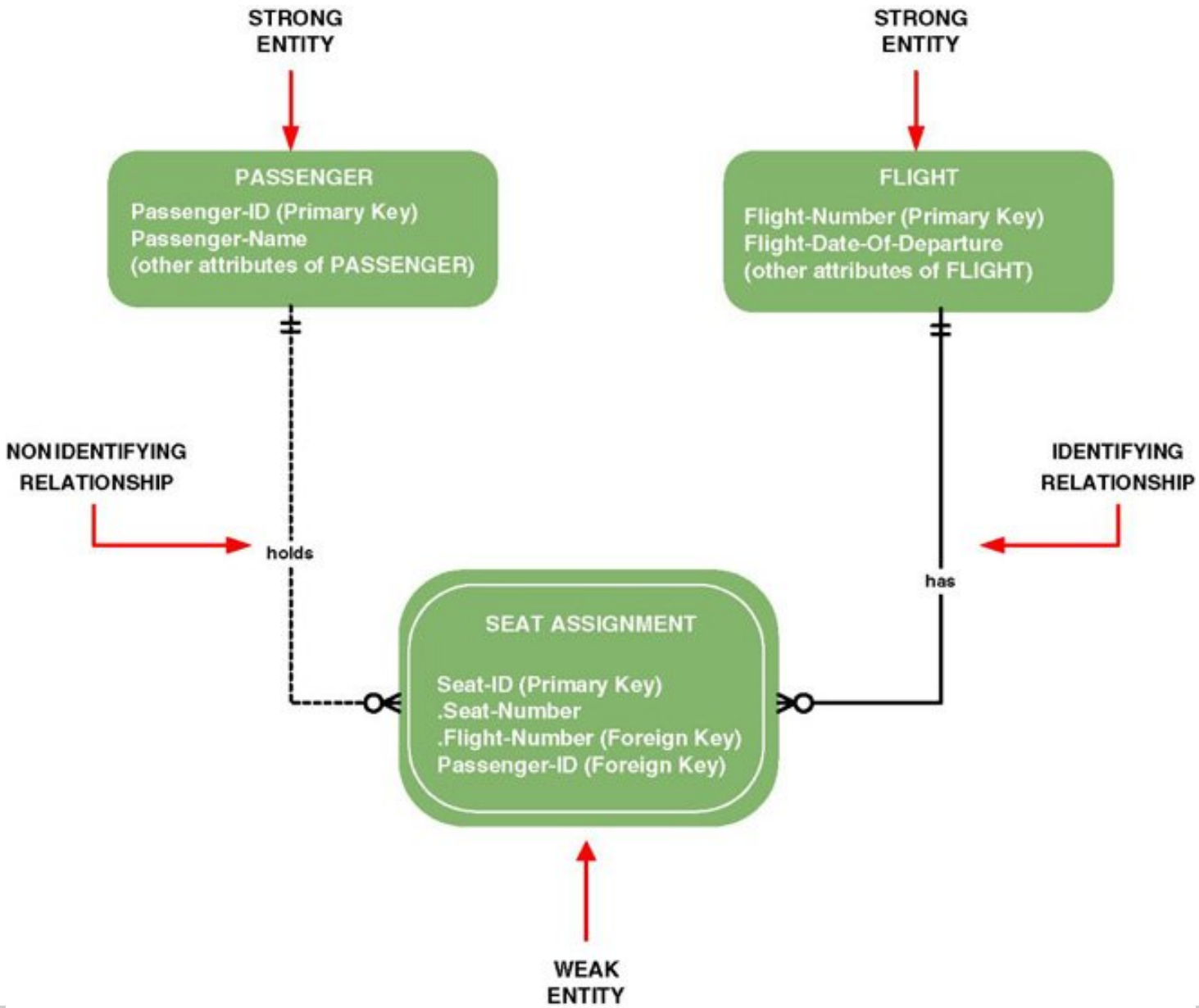
Data Modeling Concepts: Identifying Relationships

Identifying relationship – relationship in which the parent entity' key is also part of the primary key of the child entity.

- The child entity is called a *weak* entity.



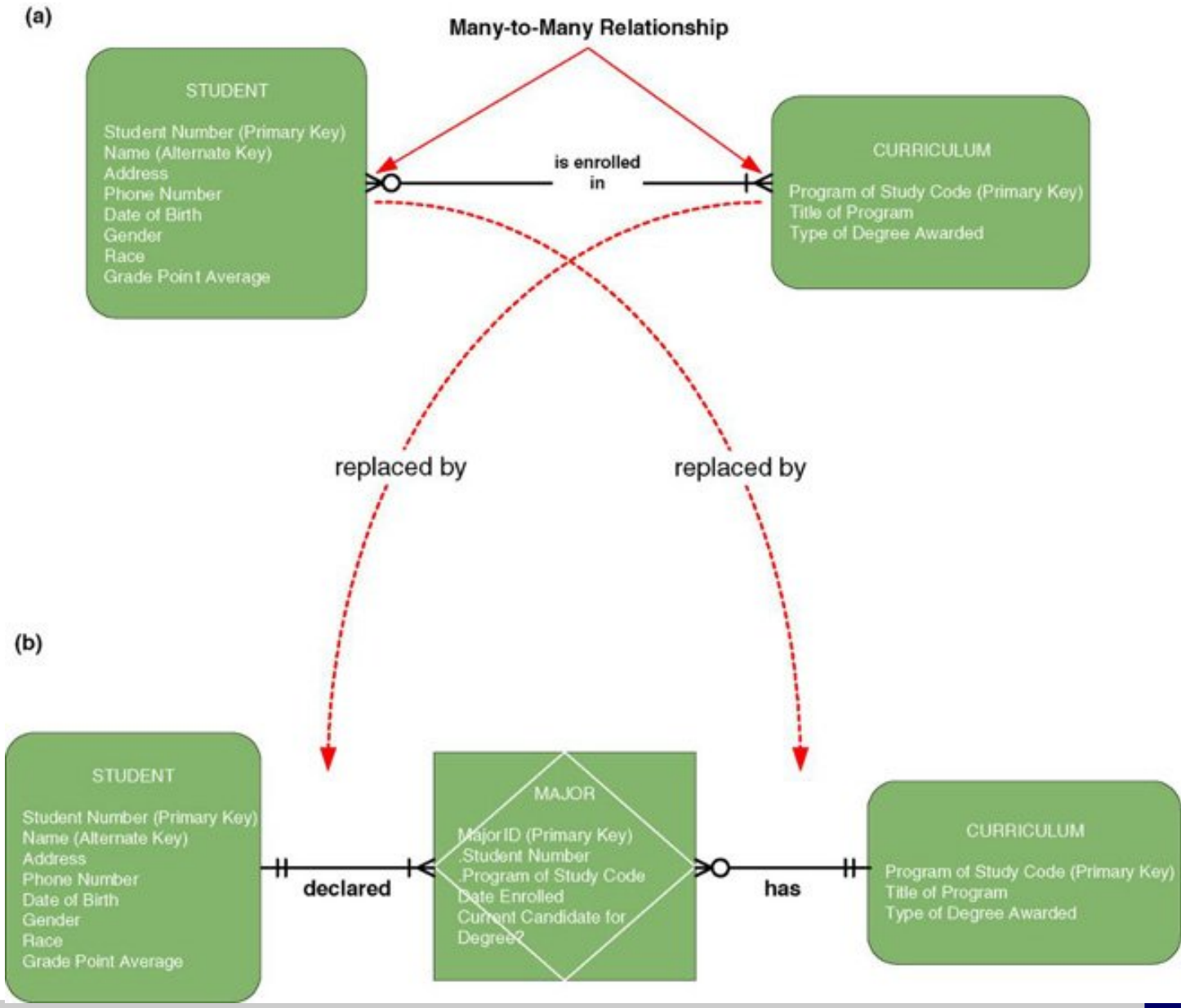
Data Modeling Concepts: Sample CASE Tool Notations



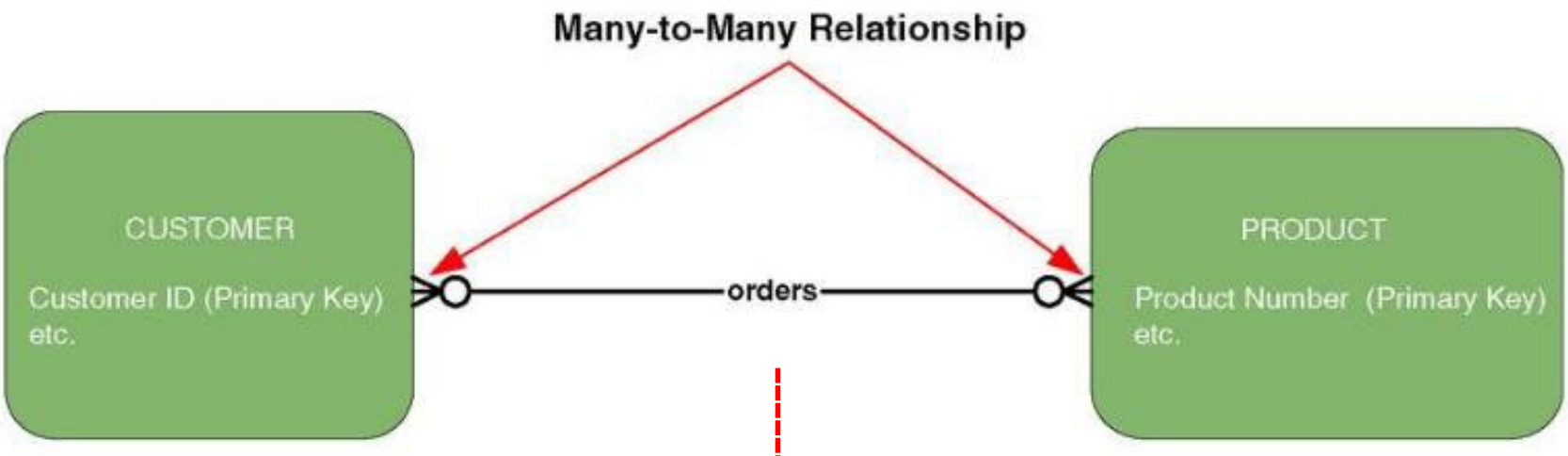
Data Modeling Concepts: Nonspecific Relationships

Nonspecific relationship – relationship where many instances of an entity are associated with many instances of another entity. Also called *many-to-many relationship*.

Nonspecific relationships must be resolved, generally by introducing an associative entity.



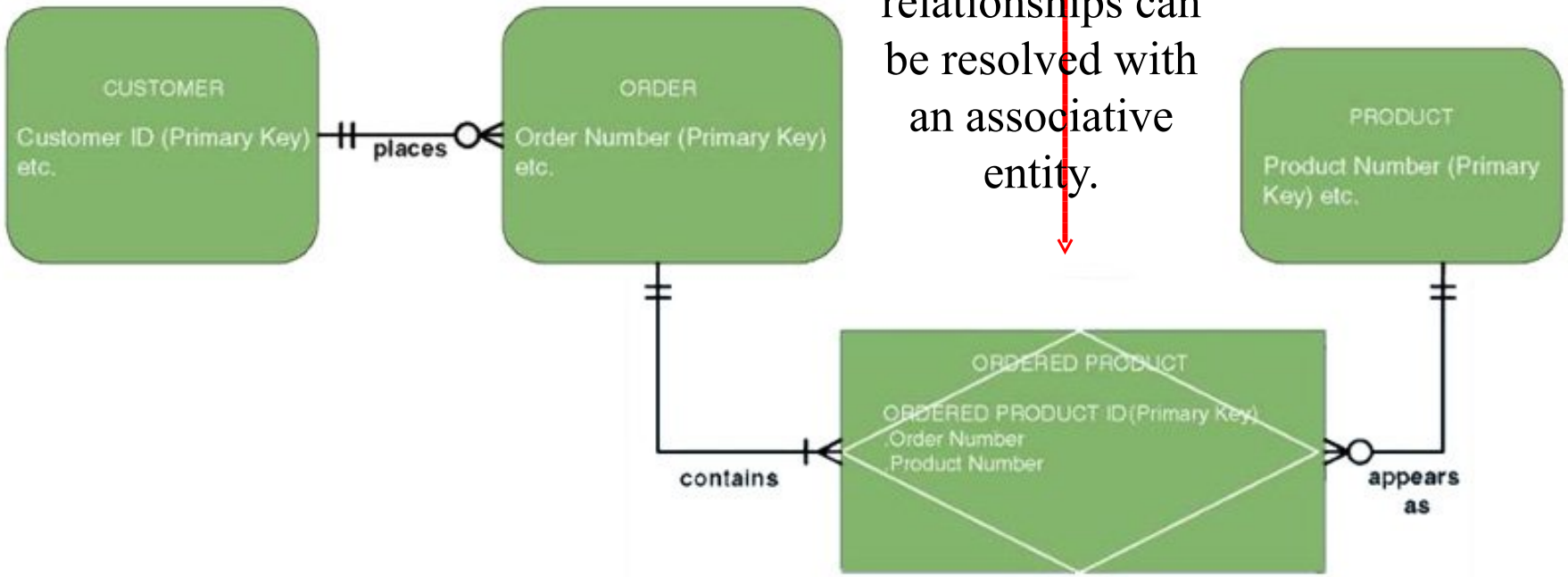
Resolving Nonspecific Relationships



The verb or verb phrase of a many-to-many relationship sometimes suggests other entities.

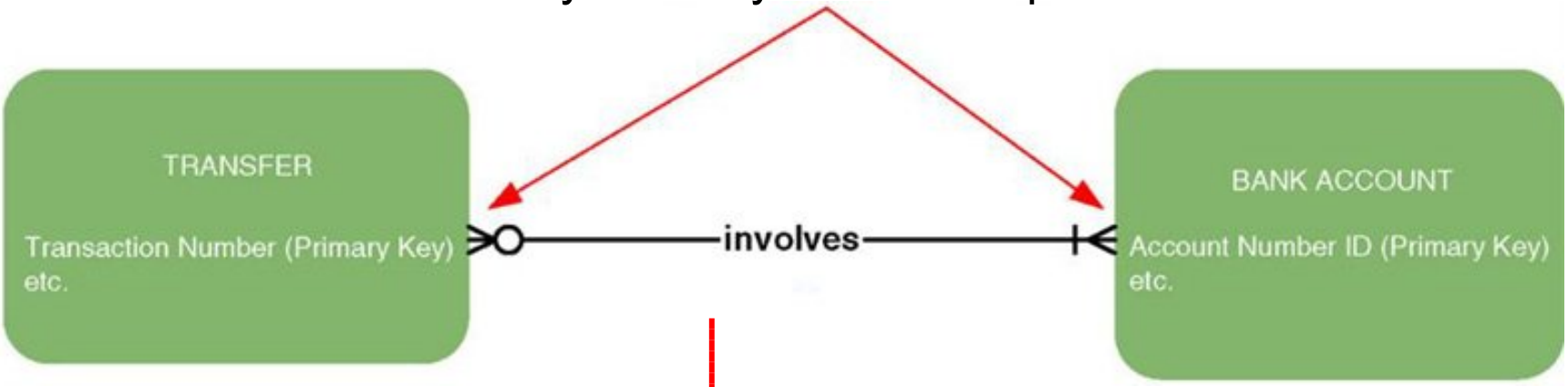
Resolving Nonspecific Relationships (continued)

Many-to-many relationships can be resolved with an associative entity.

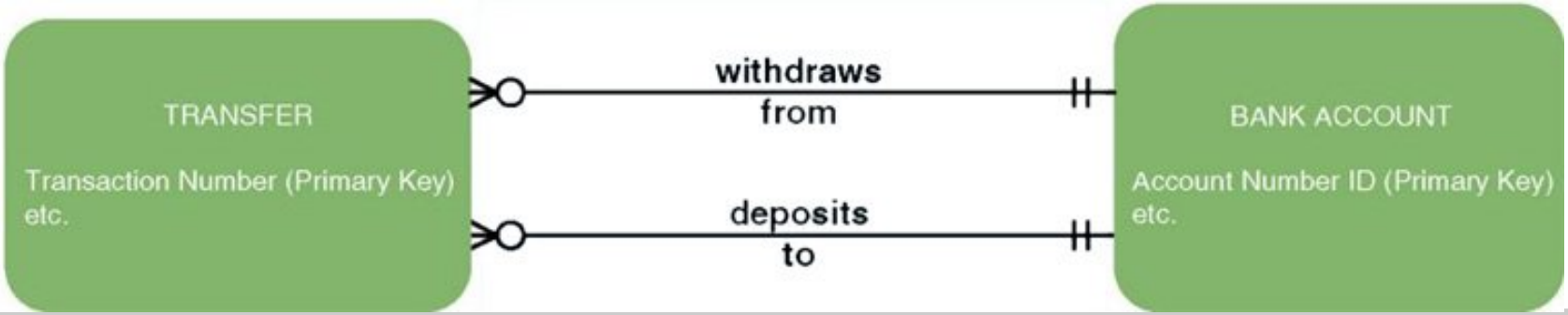


Resolving Nonspecific Relationships (continued)

Many-to-Many Relationship



While the above relationship is a many-to-many, the many on the BANK ACCOUNT side is a known maximum of "2." This suggests that the relationship may actually represent multiple relationships... In this case two separate relationships.



Data Modeling Concepts: Generalization

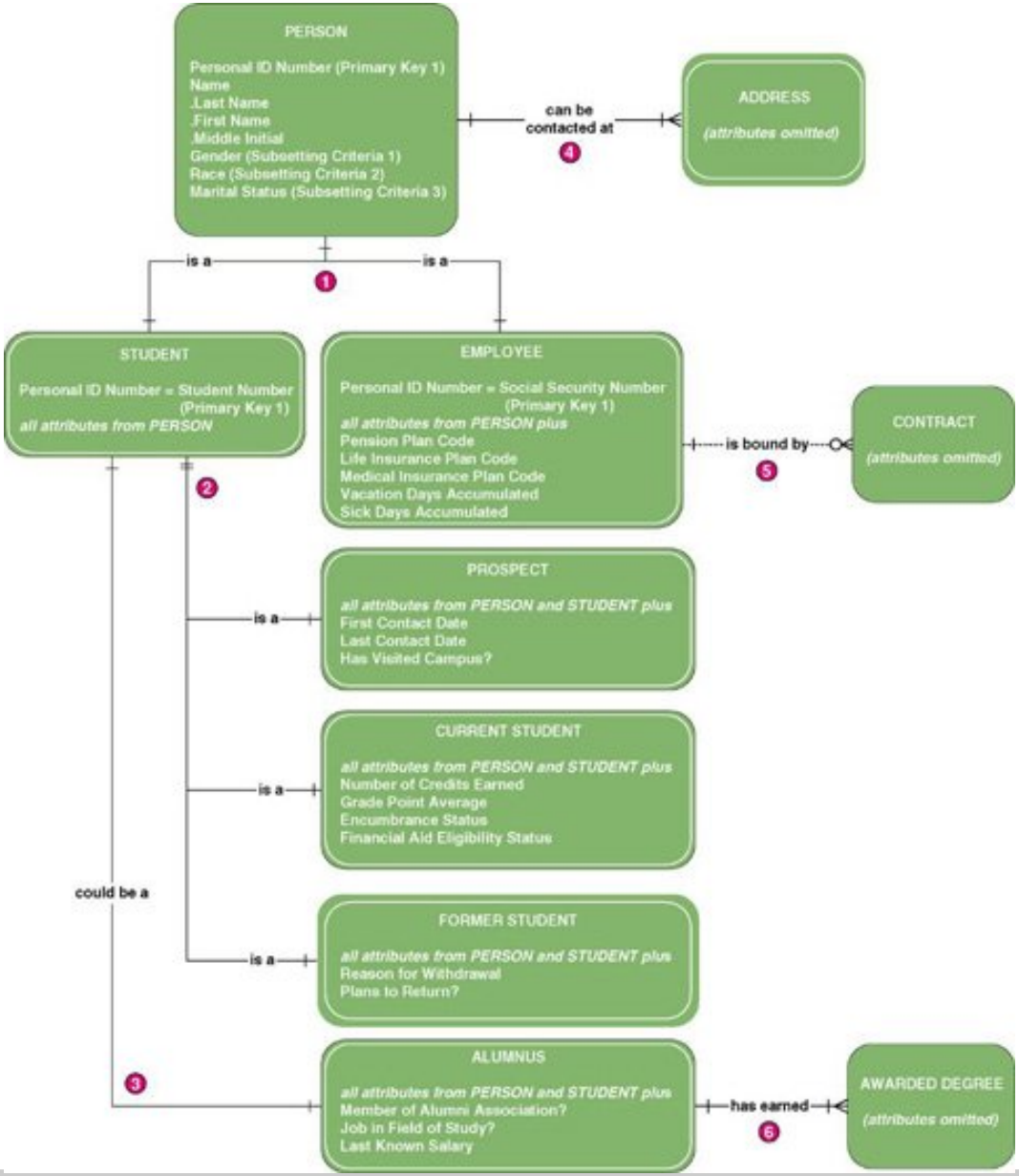
Generalization – a concept wherein the attributes that are common to several types of an entity are grouped into their own entity.

Supertype – an entity whose instances store attributes that are common to one or more entity subtypes.

Subtype – an entity whose instances may inherit common attributes from its entity supertype

And then add other attributes unique to the subtype.

Generalization Hierarchy



Process of Logical Data Modeling

- Strategic Data Modeling
  - Many organizations select IS development projects based on strategic plans.
    - Includes vision and architecture for information systems
    - Identifies and prioritizes develop projects
    - Includes enterprise data model as starting point for projects
- Data Modeling during Systems Analysis
  - Data model for a single information system is called an application data model.

Logical Model Development Stages

1. Context Data model
   - Includes only entities and relationships
   - To establish project scope
2. Key-based data model
   - Eliminate nonspecific relationships
   - Add associative entities
   - Include primary and alternate keys
   - Precise cardinalities
3. Fully attributed data model
   - All remaining attributes
   - Subsetting criteria
4. Normalized data model

Metadata - data about data.

JRP and Interview Questions for Data Modeling

Purpose	Candidate Questions (see textbook for a more complete list)
Discover system entities	What are the subjects of the business?
Discover entity keys	What unique characteristic (or characteristics) distinguishes an instance of each subject from other instances of the same subject?
Discover entity subsetting criteria	Are there any characteristics of a subject that divide all instances of the subject into useful subsets?
Discover attributes and domains	What characteristics describe each subject?
Discover security and control needs	Are there any restrictions on who can see or use the data?
Discover data timing needs	How often does the data change?
Discover generalization hierarchies	Are all instances of each subject the same?
Discover relationships?	What events occur that imply associations between subjects?
Discover cardinalities	Is each business activity or event handled the same way, or are there special circumstances?

Automated Tools for Data Modeling

SA/2001 - H:\SOUNDSTAGE SA ENCYCLOPEDIA - 3NF VERSION\ - [3NF Member Services (Entity Relation Subject Area)]

File Edit View Format Draw Tools Dictionary Reports Window Help

Browse

Diagrams

Data Fl

Decomp

Entity R

3NF M

Memb

The S

MEMBER ORDERED PRODUCT
-Primary Key
Order-Number [PK1] [FK]
Product-Number [PK2] [FK]
-Non-Key Attributes
Quantity-Ordered
Quantity-Shipped
Quantity-Backordered

sells

MEMBER ORDER
-Primary Key
Order-Number [PK1]
-Non-Key Attributes
Order-Creation-Date
Order-Fill-Date
Shipping-Address-Name
Shipping-Street-Address
Shipping-City
ping-State
ping-Zip
ping-Instructions
er-Sub-Total
er-Sales-Tax
er-Shipping-Method
er-Shipping-&-Handling-Cost
er-Status
er-Prepaid-Amount
er-Prepayment-Method
notion-Number [FK]
ber-Number [FK]
ber-Number-1 . Member-Number [FK]

responds to

places

Model Object - Entity - MEMBER ORDER

Name MEMBER ORDER

Attributes | Entity Info. | Access Paths | Check Constraint | Triggers & Table Segment | Source Info. | Symbol

Attribute List

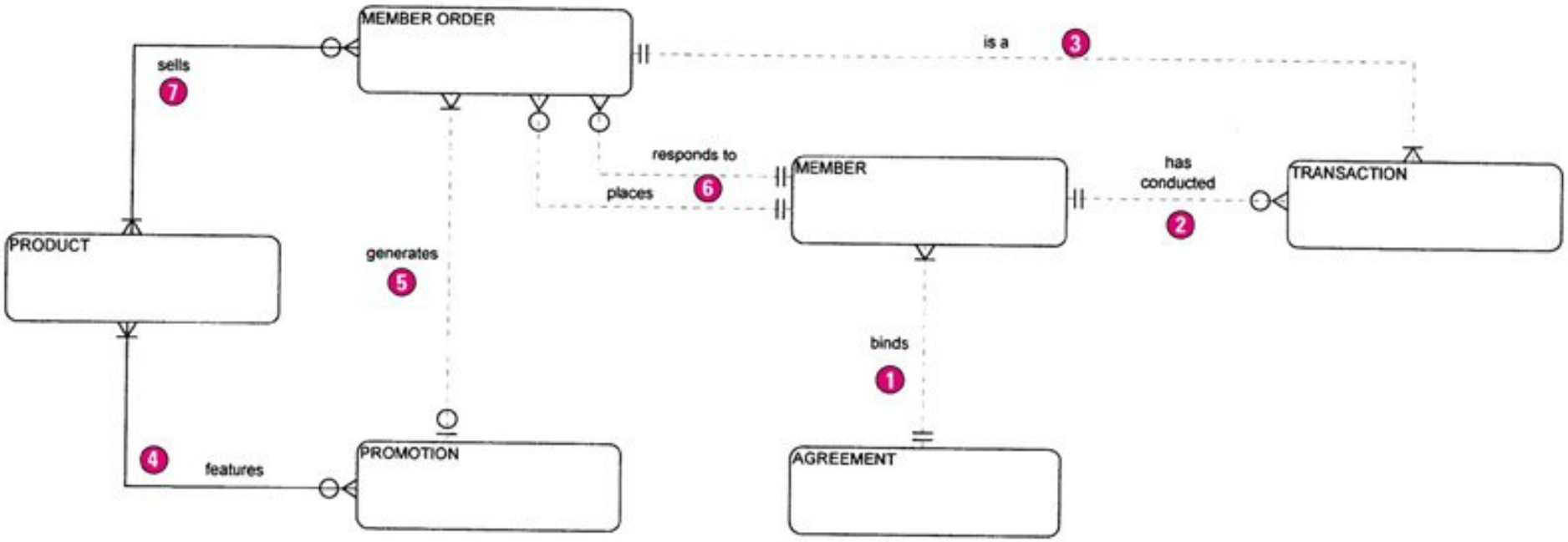
	Data	Name	PK	FK	Allow Null	Unique	Data Type	Qualifie
1	Order-Number	Order-Number	☑	☐	☐	☐	integer	
2	Order-Creation-	Order-Creation-D	☐	☐	☐	☐	smalldate	
3	Order-Fill-Date	Order-Fill-Date	☐	☐	☑	☐	smalldate	
4	Shipping-Addre	Shipping-Address	☐	☐	☑	☐	character	(40)
5	Shipping-Street-	Shipping-Street-A	☐	☐	☐	☐	character	(40)
6	Shipping-City	Shipping-City	☐	☐	☐	☐	character	(30)
7	Shipping-State	Shipping-State	☐	☐	☐	☐	character	(2)

Insert | Delete | Define | Choices...

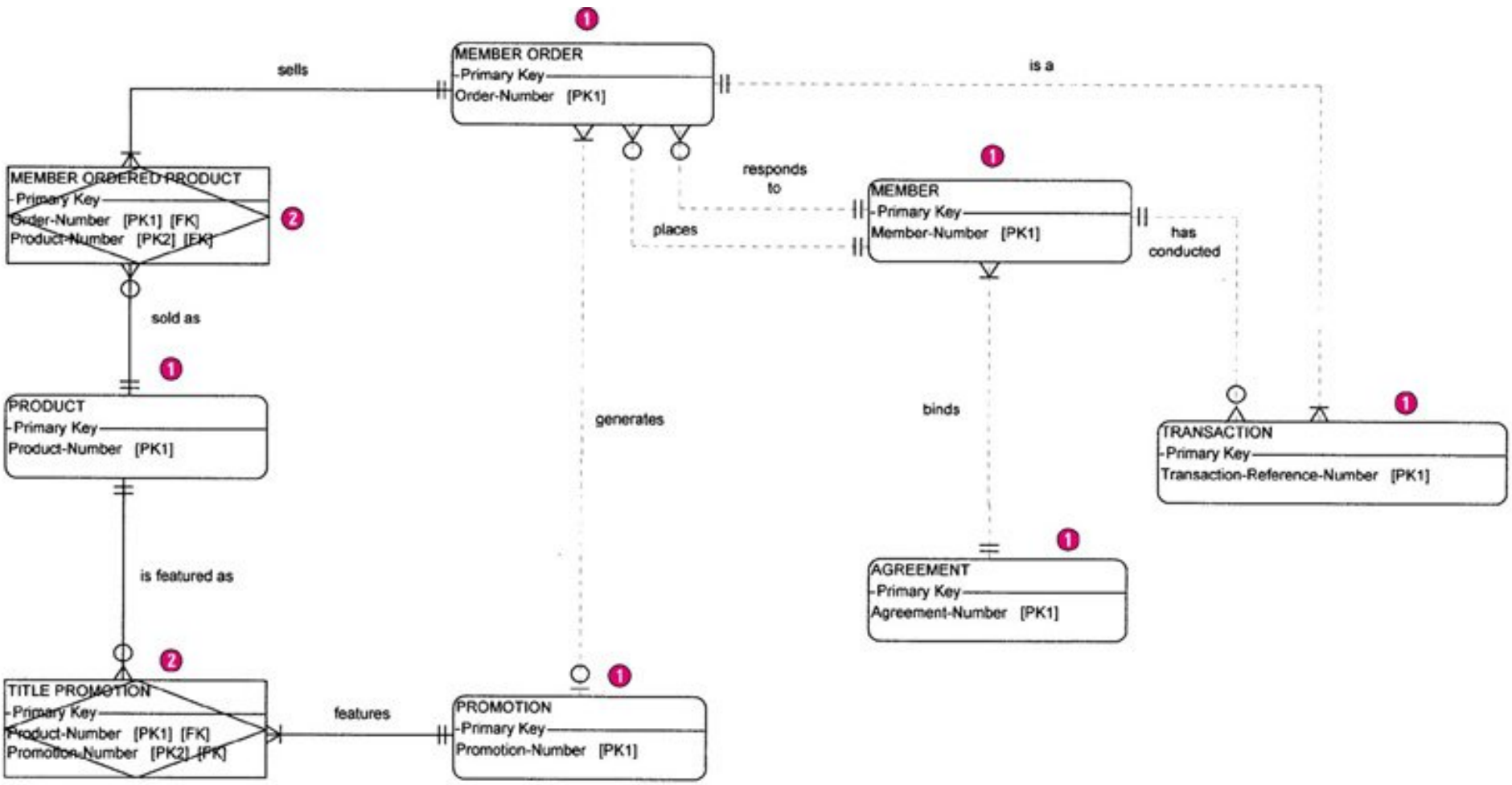
OK | Cancel | Spell | Delete

MEMBER ORDER | 5.42", 0.11" | 3.19" x 4.42" | Idb

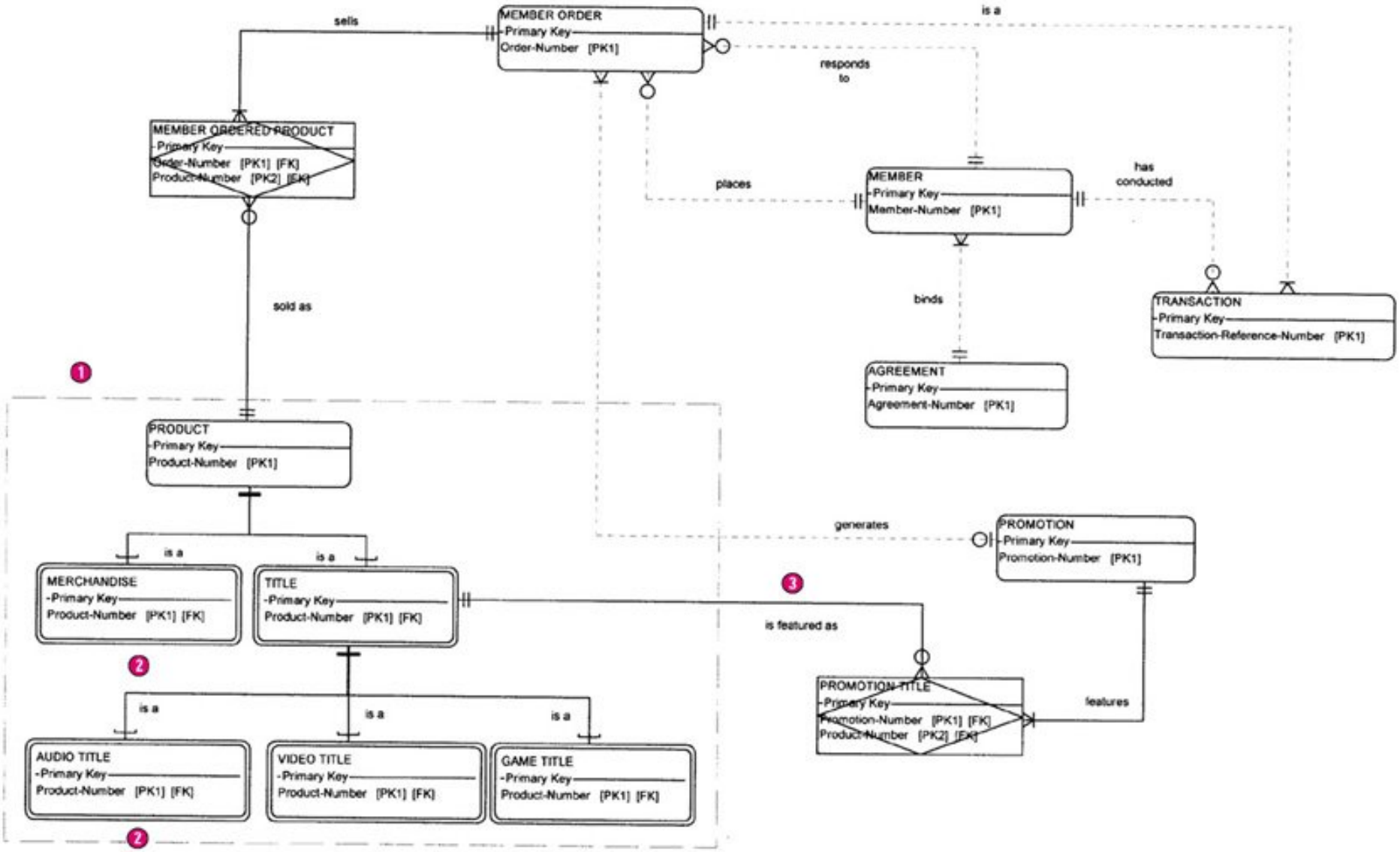
The Context Data Model



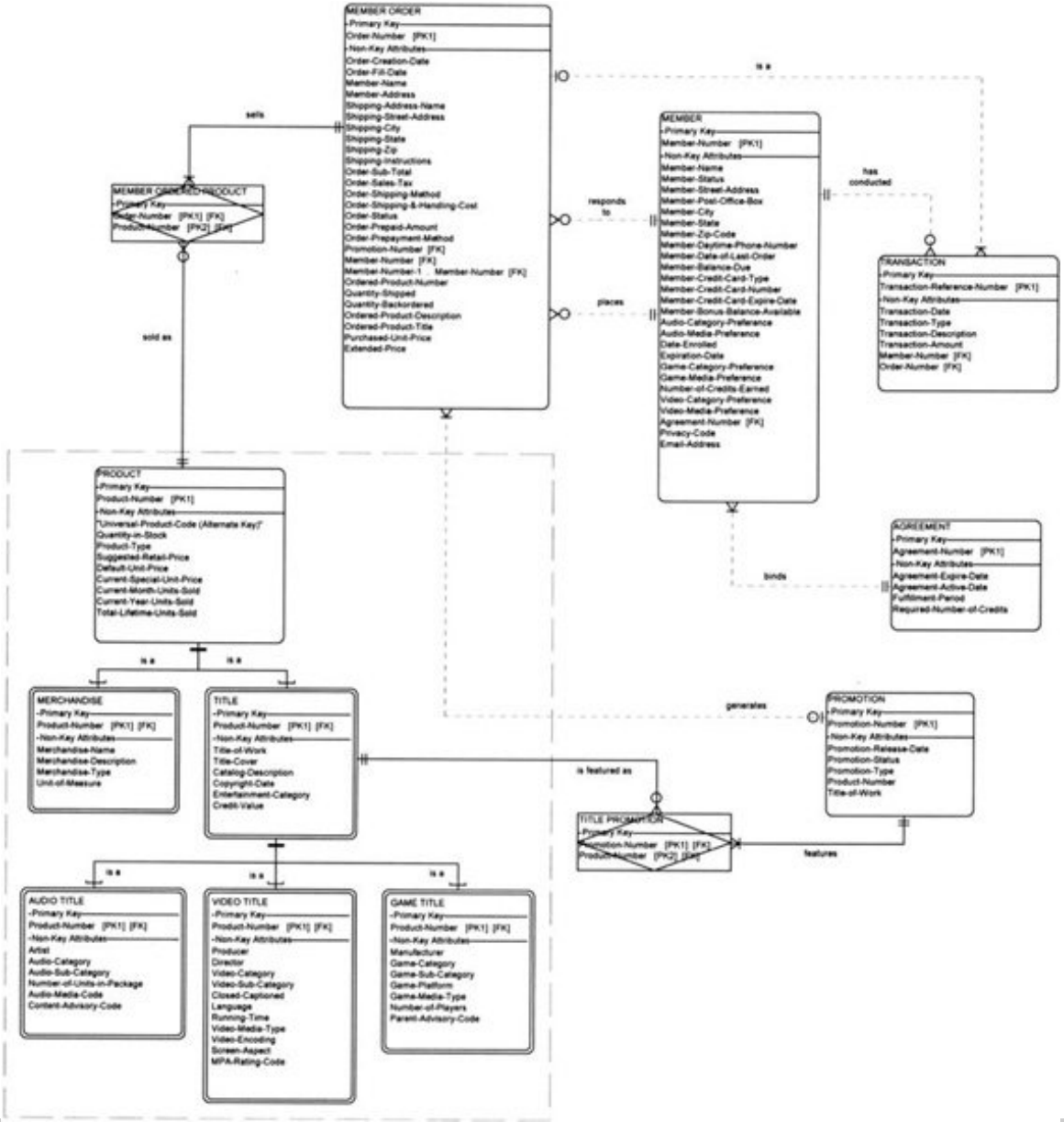
The Key-based Data Model



The Key-based Data Model with Generalization



The Fully-Attributed Data Model



What is a Good Data Model?

- A good data model is simple.
  - Data attributes that describe any given entity should describe only that entity.
  - Each attribute of an entity instance can have only one value.
- A good data model is essentially nonredundant.
  - Each data attribute, other than foreign keys, describes at most one entity.
  - Look for the same attribute recorded more than once under different names.
- A good data model should be flexible and adaptable to future needs.

Data Analysis & Normalization

Data analysis – a technique used to improve a data model for implementation as a database.

- Goal is a simple, nonredundant, flexible, and adaptable database.

Normalization – a data analysis technique that organizes data into groups to form nonredundant, stable, flexible, and adaptive entities.

Normalization: 1NF, 2NF, 3NF

First normal form (1NF) – entity whose attributes have no more than one value for a single instance of that entity

- Any attributes that can have multiple values actually describe a separate entity, possibly an entity and relationship.

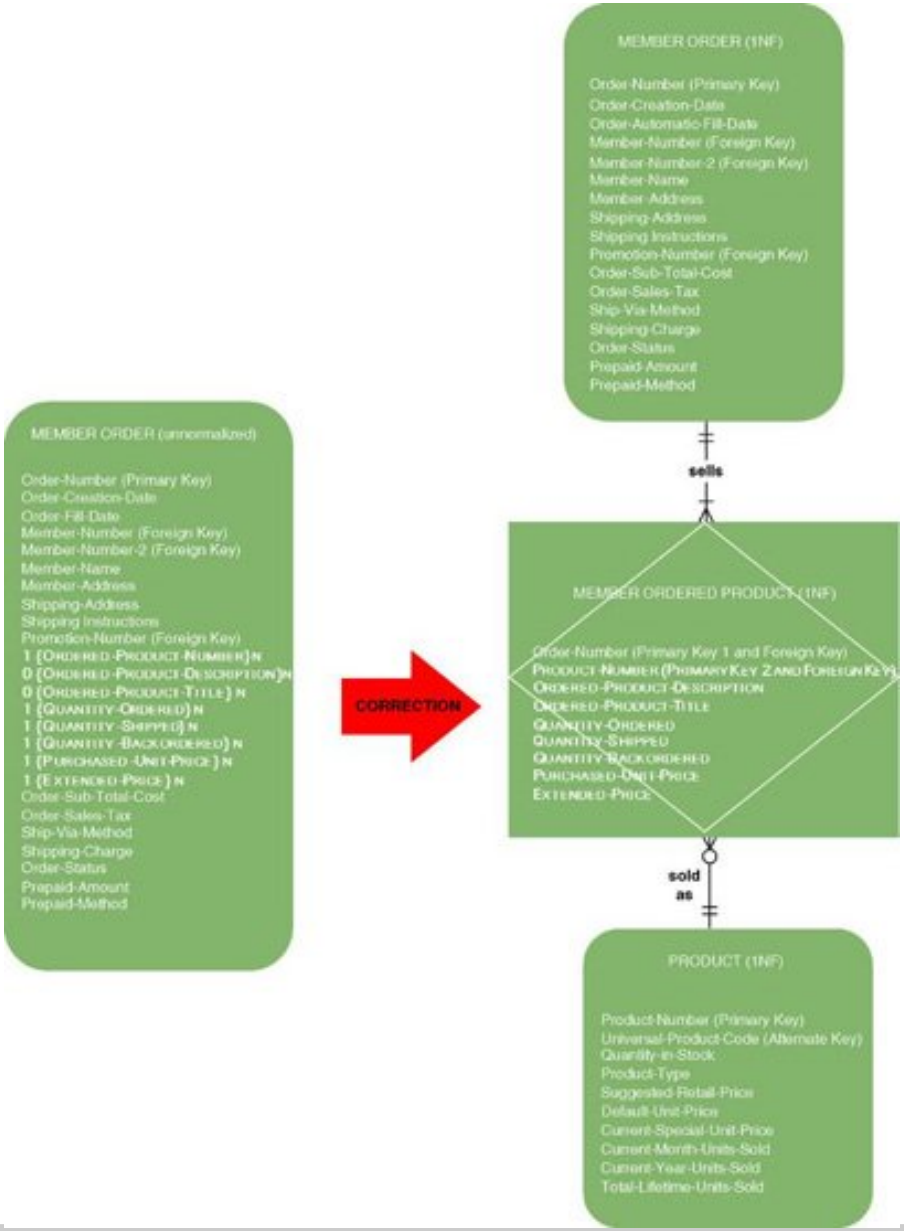
Second normal form (2NF) – entity whose nonprimary-key attributes are dependent on the full primary key.

- Any nonkey attributes dependent on only part of the primary key should be moved to entity where that partial key is the full key. May require creating a new entity and relationship on the model.

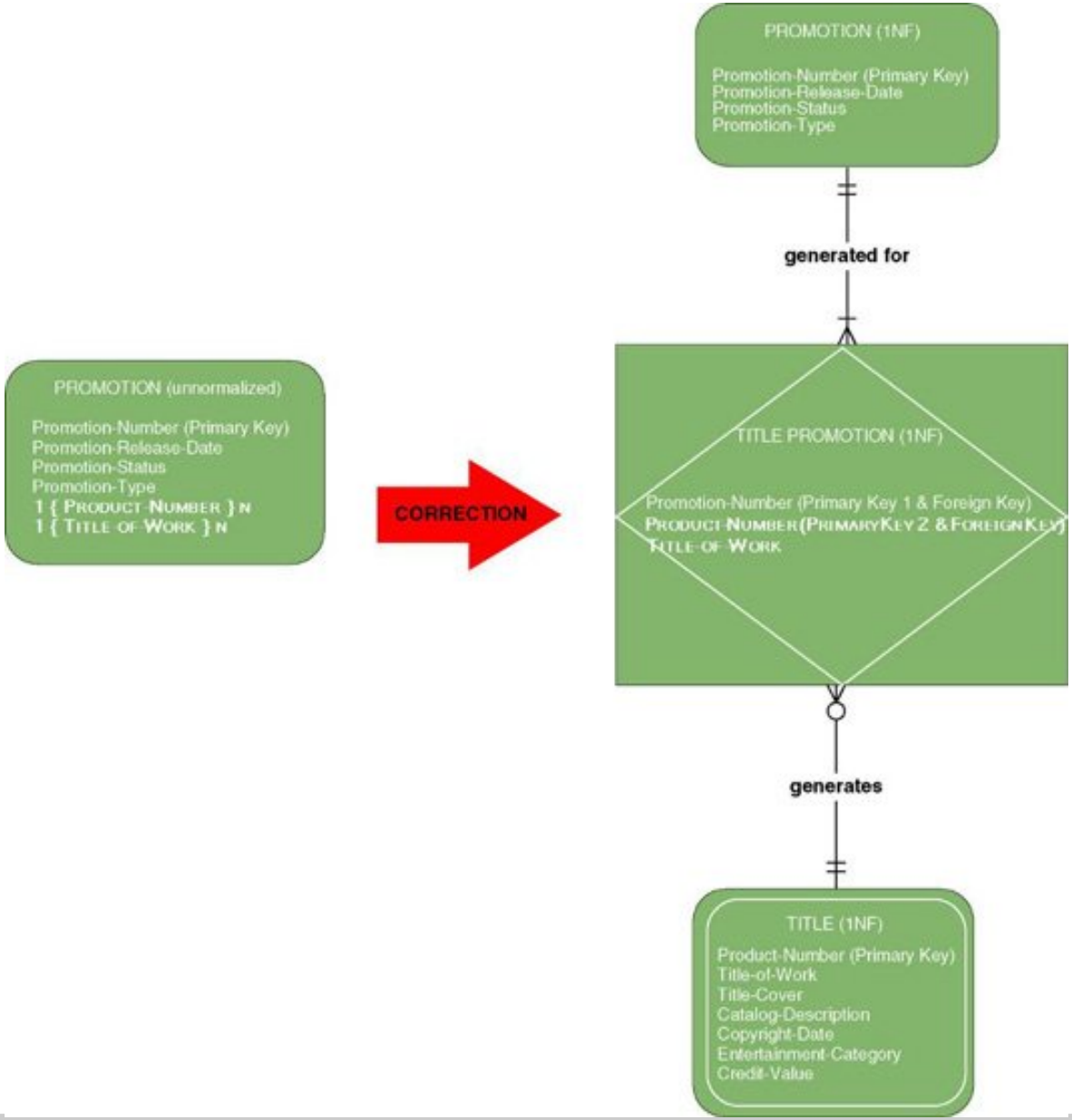
Third normal form (3NF) – entity whose nonprimary-key attributes are not dependent on any other non-primary key attributes.

- Any nonkey attributes that are dependent on other nonkey attributes must be moved or deleted. Again, new entities and relationships may have to be added to the data model.

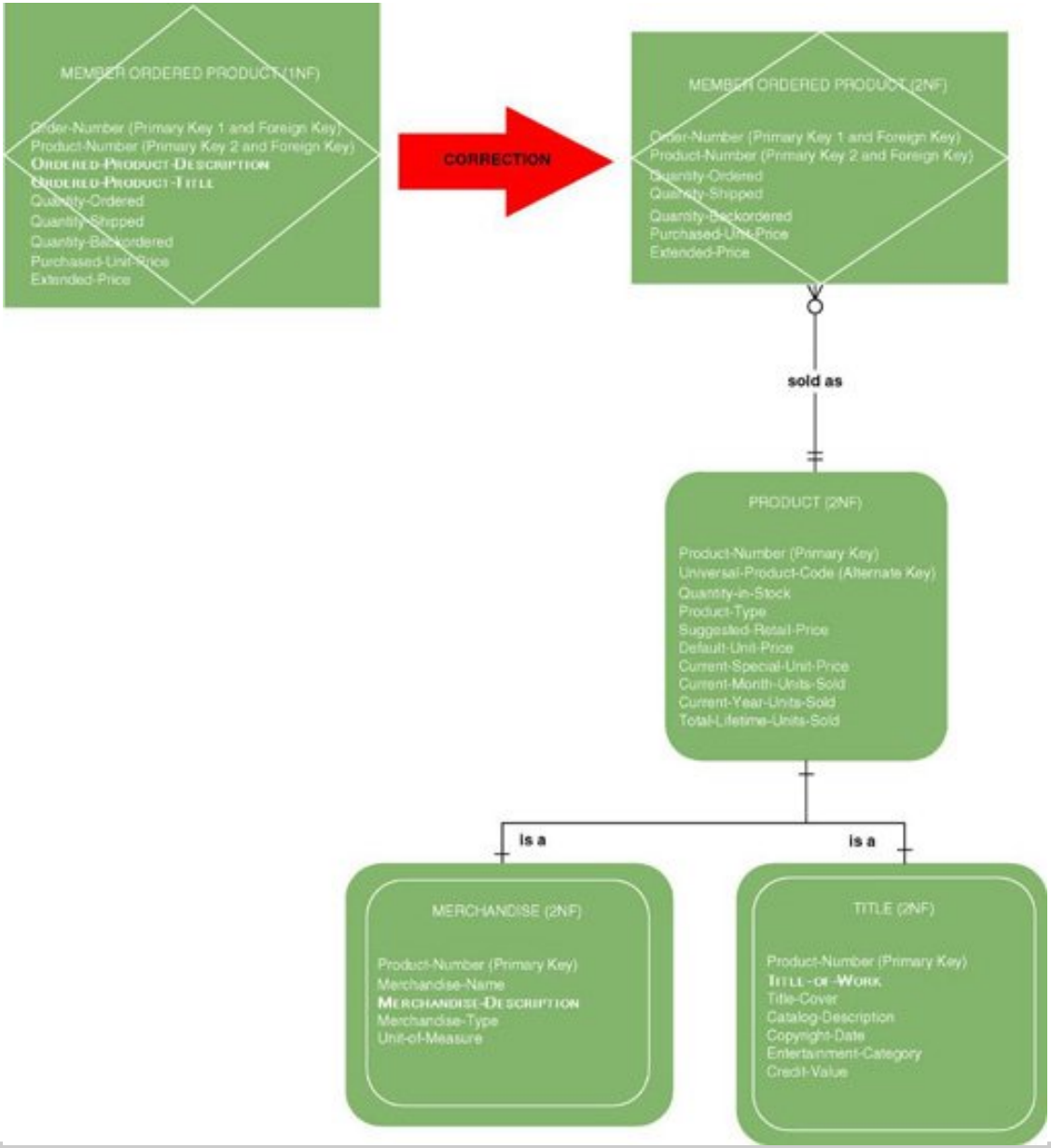
First Normal Form Example 1



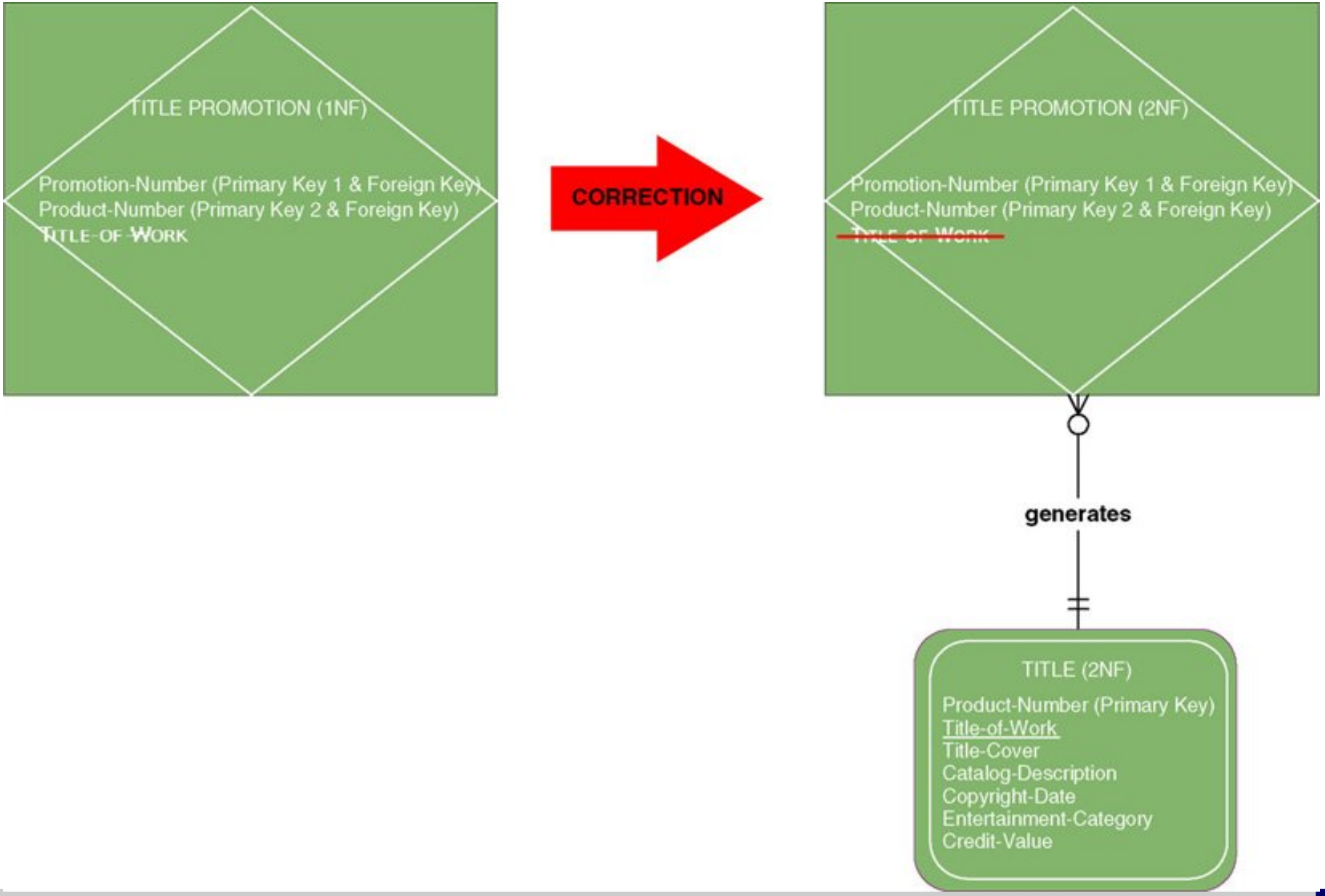
First Normal Form Example 2



Second Normal Form Example 1



Second Normal Form Example 2

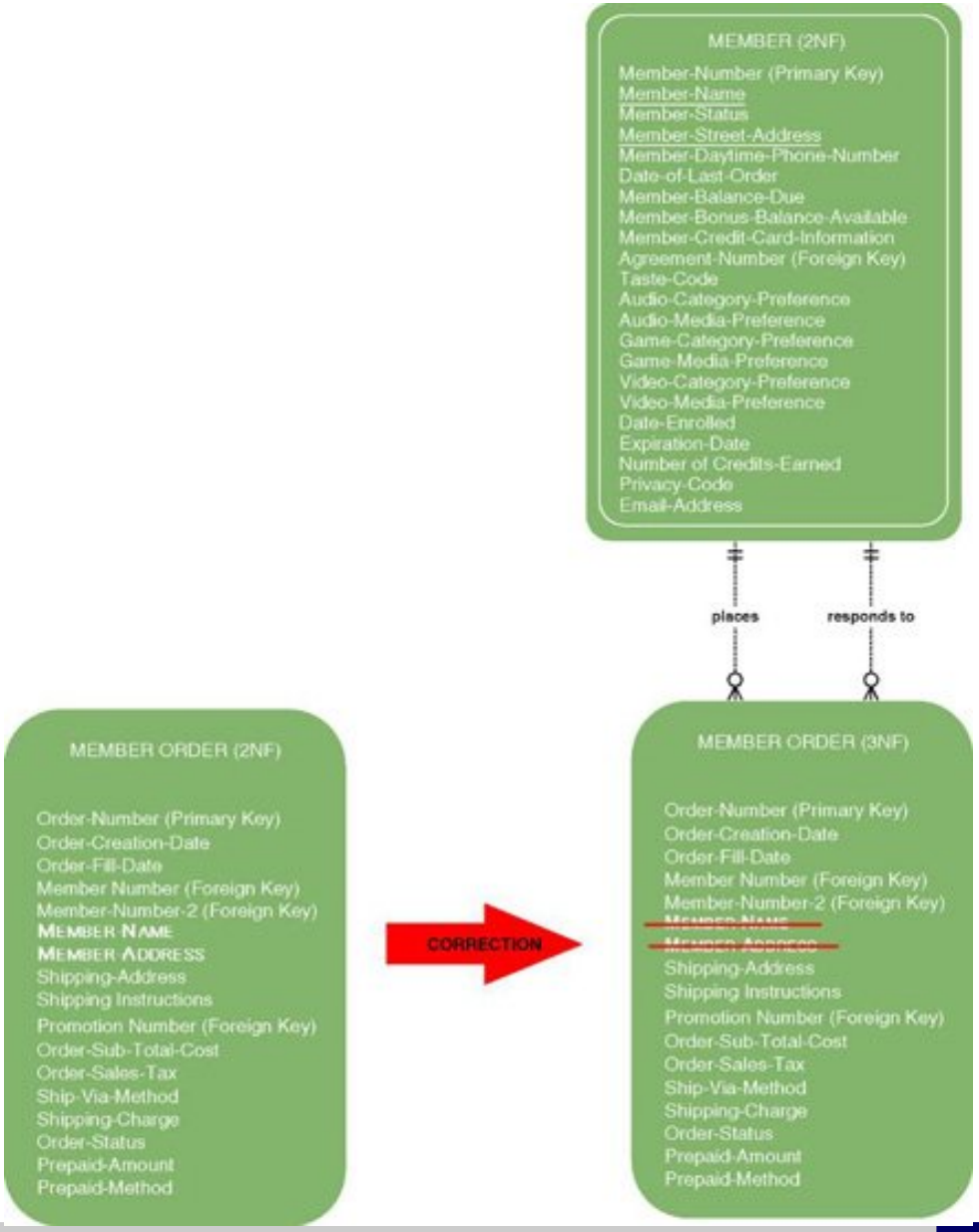


Third Normal Form Example 1

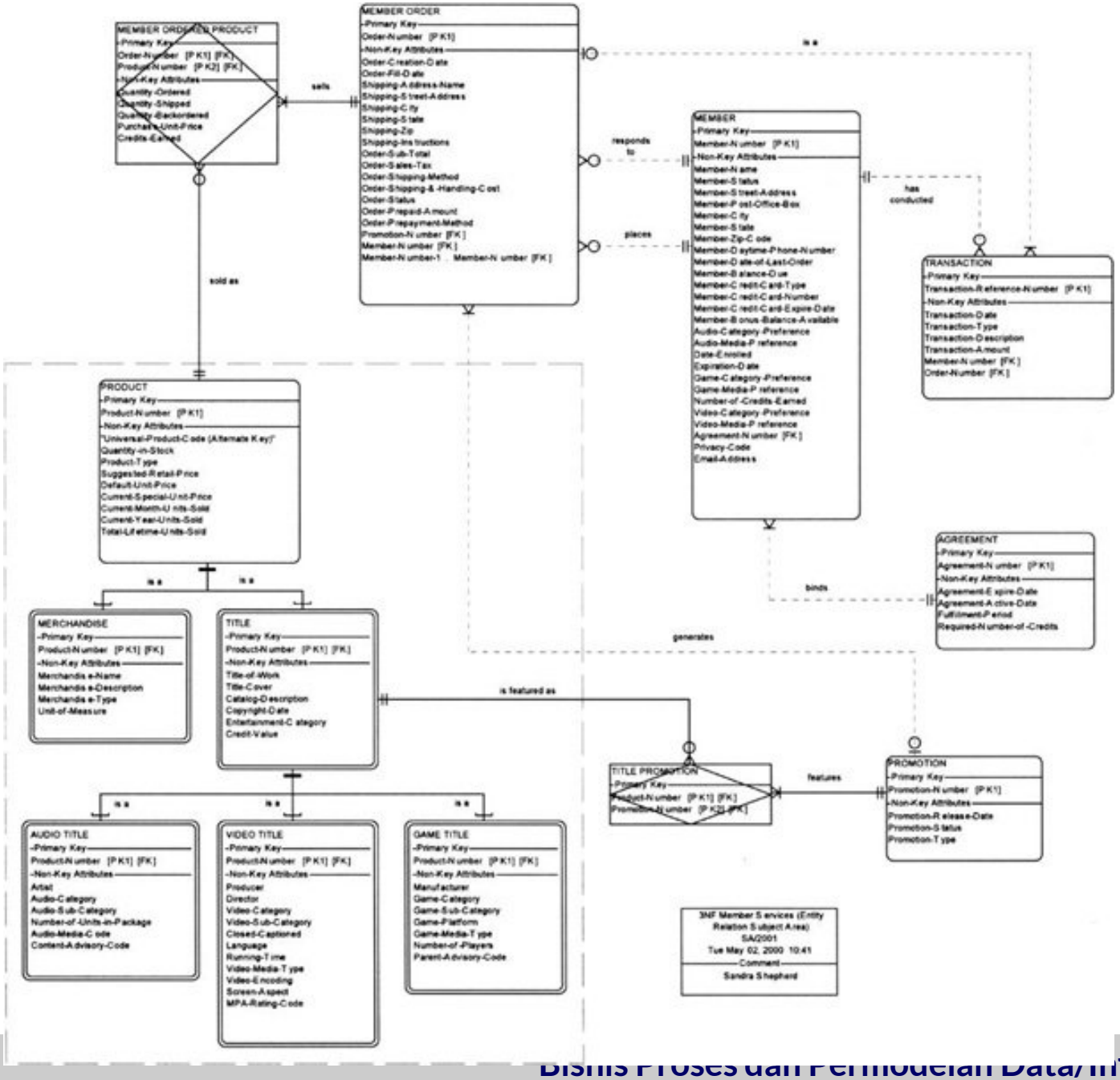
Derived attribute – an attribute whose value can be calculated from other attributes or derived from the values of other attributes.

Third Normal Form Example 2

Transitive dependency – when the value of a nonkey attribute is dependent on the value of another nonkey attribute other than by derivation.



SoundStage 3NF Data Model



Data-to-Location-CRUD Matrix

Entity . Attribute / Location	Customers	Kansas City	. Marketing	. Advertising	. Warehouse	. Sales	. A/R	Boston	. Sales	. Warehouse	San Francisco	. Sales	San Diego	. Warehouse
Customer	**INDV**					**ALL**	**ALL**		**SS**	**SS**		**SS**		**SS**
.Customer Number	R				R	CRUD	R		CRUD	R		CRUD		R
.Customer Name	RU				R	CRUD	R		CRUD	R		CRUD		R
.Customer Address	RU				R	CRUD	R		CRUD	R		CRUD		R
.Customer Credit Rating	X					R	RU		R			R		
.Customer Balance Due	R					R	RU		R			R		
Order	**INDV**		**ALL**		**SS**	**ALL**			**SS**	**SS**		**SS**		**SS**
.Order Number	SRD		R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
.Order Date	SRD		R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
.Order Amount	SRD		R	CRUD		CRUD	R		CRUD	R		CRUD		R
Ordered Product	**INDV**		**ALL**		**SS**	**ALL**			**SS**	**SS**		**SS**		**SS**
.Quantity Ordered	SUD		R	CRUD	R	CRUD	R		CRUD			CRUD		
.Ordered Item Unit Price	SUD		R	CRUD		CRUD	R		CRUD			CRUD		
Product	**ALL**		**ALL**	**ALL**	**ALL**	**ALL**			**ALL**	**ALL**		**ALL**		**ALL**
.Product Number	R		CRUD	R	R	R			R	R		R		R
.Product Name	R		CRUD	R	R	R			R	R		R		R
.Product Description	R		CRUD	RU	R	R			R	R		R		R
.Product Unit of Measure	R		CRUD	R	R	R			R	R		R		R
.Product Current Unit Price	R		CRUD	R		R			R	R		R		R
.Product Quantity on Hand	X				RU	R			R	RU		R		RU

INDV = individual **ALL** = ALL **SS** = subset X = no access

S = submit C = create R = read U = update D = delete

Attribute Meta-Data

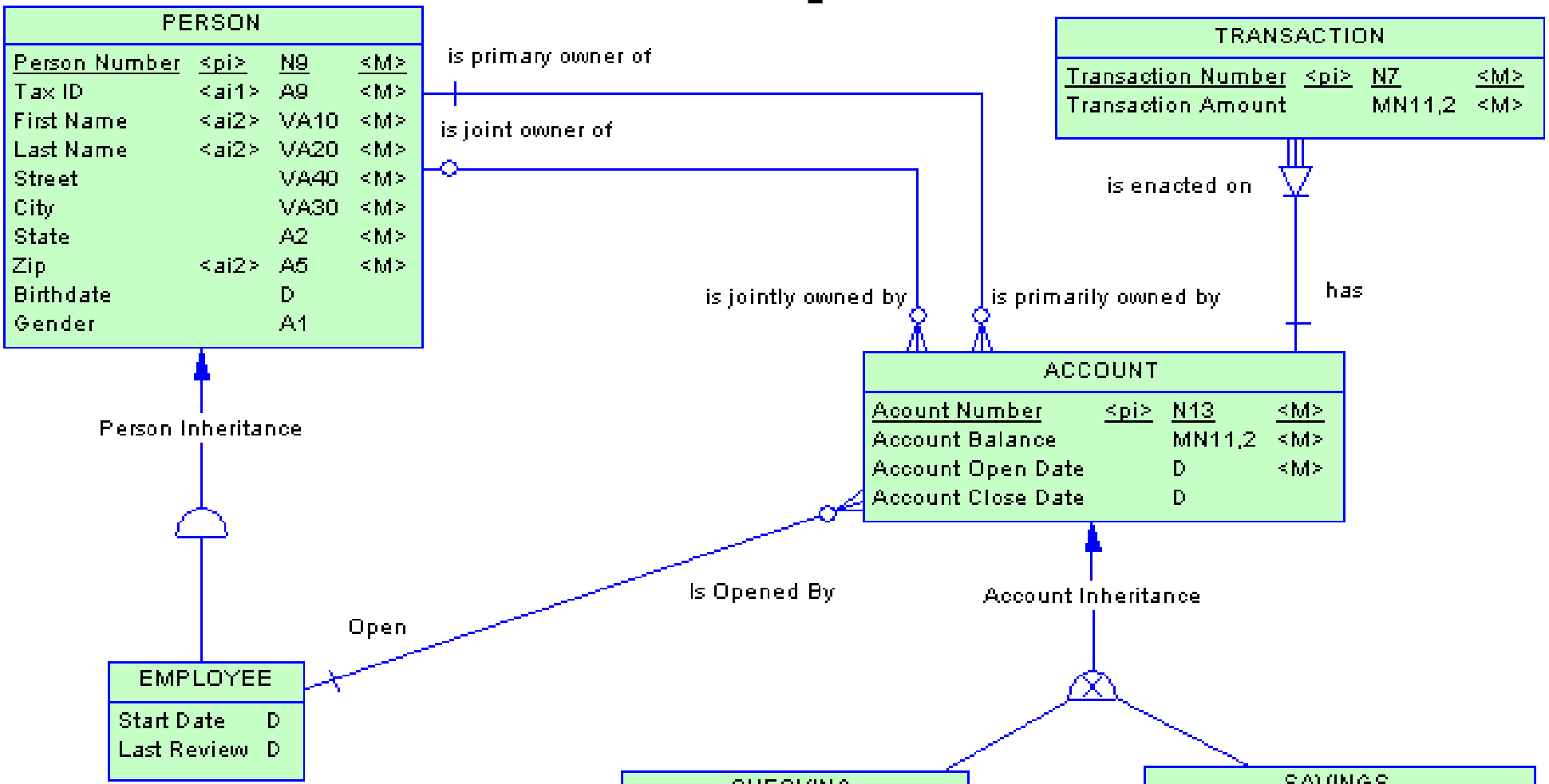
- Comment
- Business Data Type
- Optional or Mandatory
- Part of Unique Identifier(s)
- Checks
  - Minimum
  - Maximum
  - Default
  - Units
  - Format
  - Upper/Lowercase
  - Cannot Modify
  - List of Legal Values
  - Business Rules
- NOTE - This represents the bulk of the work when data modeling

Failure to discover & document attribute meta-data is where many data models fail

Failure to capture this meta-data results in ineffective databases: that's where bad data is allowed

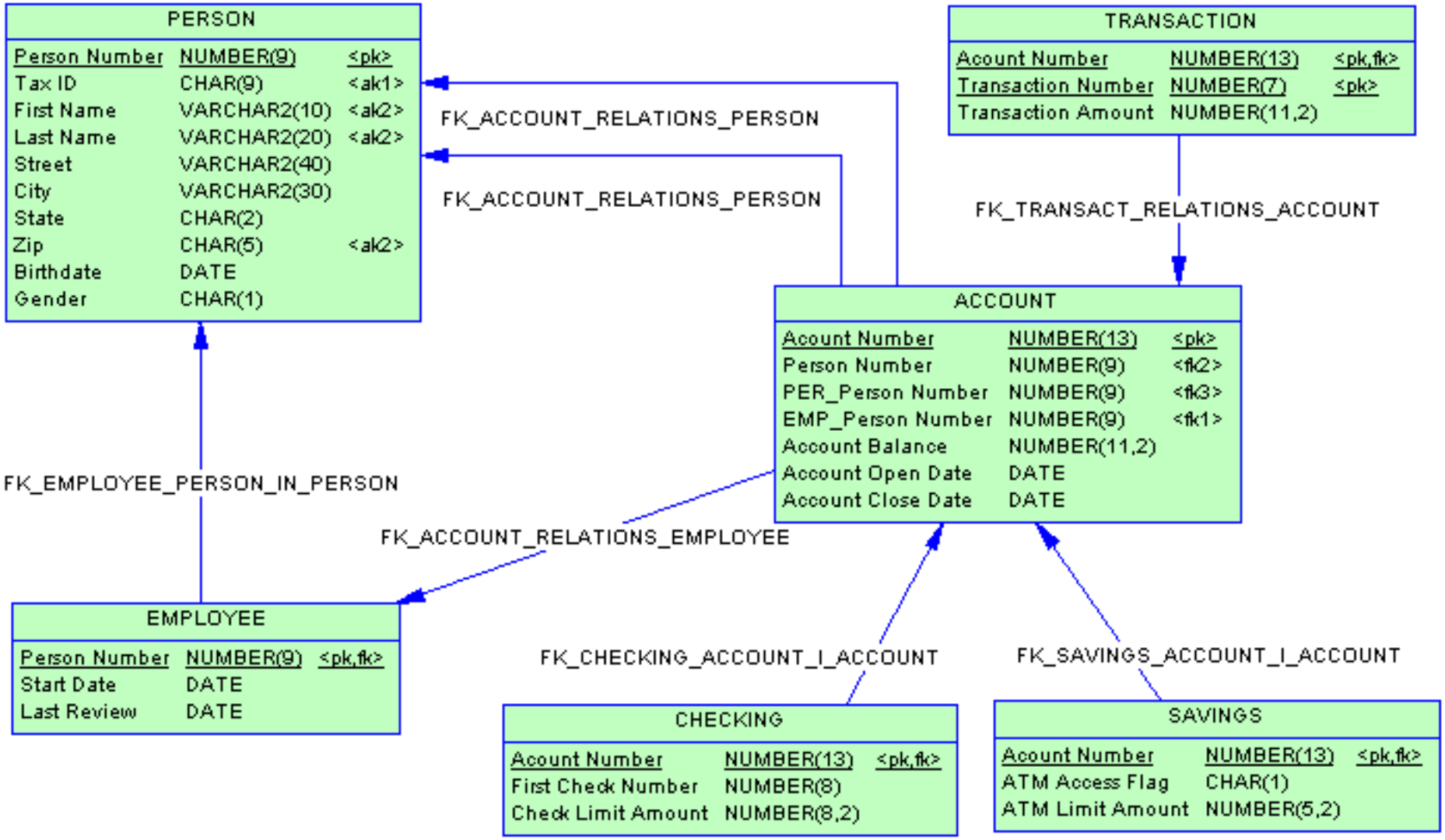
Next few slides show Quest's QDesigner – other modeling tools will have similar screens

Final Conceptual Model

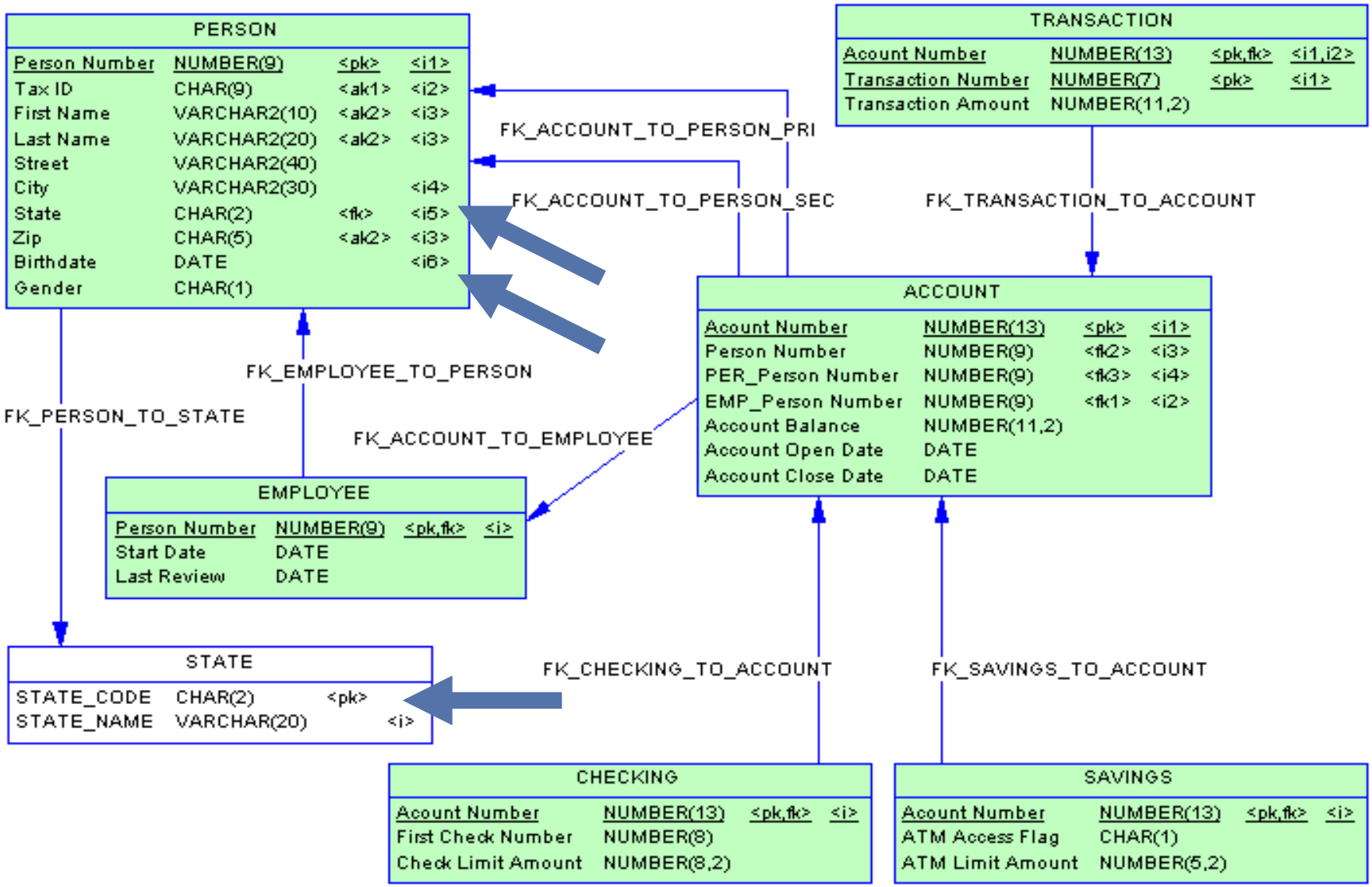


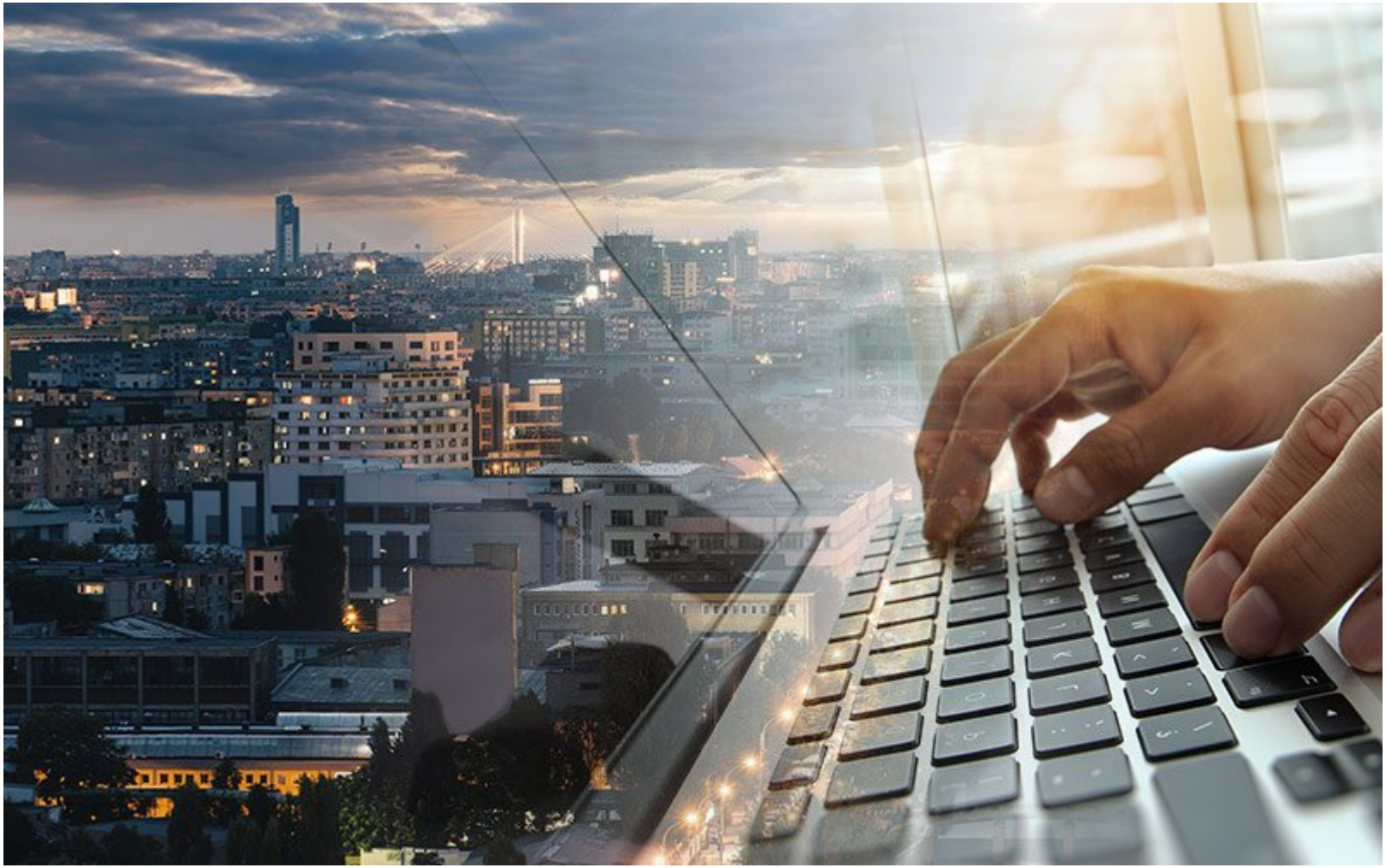
Employee's can be Customers – right?

Initial Physical Model



Final Physical Model





Kamus Data dan Metadata

Kamus Data

- Bukan sekedar
  - Data, Jenis Data
  - Memiliki informasi lebih lengkap misal domain, kodifikasi dan sebagianya
- Metadata
  - Data tentang data
  - Lebih lengkap lagi untuk penjejakan, interoperabilitas dsb

Untuk setiap dataset ada info:

- Basic dataset introduction must include:
- Who collected or aggregated that data, or in the case of many contributors, who is the principal investigator or contact.
- When the data was collected.
- What the data elements are measuring or describing.
- Why the data was collected.
- Methodologies used or assumptions made while collecting the data

- Additional definition should include when relevant:
  - Description of any transformations or calculations applied to the raw data (if the data being described is not the raw data) or to specific data elements, including references to any scripts used.
  - Versionv.
  - Any validation or quality control process that has been applied.

Deskripsi Elemen Data

- Element name as found in the dataset, i.e., the data label, column header, or filename.
- A full “human readable” name of the element if the dataset uses codes or abbreviations.
- A definition of the data element.
- Any units of measure and precision (if applicable), e.g., “measured in meters, rounded up to the nearest .01 meter”.
- The format of the data element (if applicable), such as integer, text, date-time, etc.
- All valid/allowed values.
- Any codes, symbols or abbreviations used in the values themselves.
- If the element is required or not.
- The source of the controlled vocabulary or thesaurus used (if applicable).
- The source of the data element, e.g., sensor, observation, etc.
- If the element is “null”(for a non-required element) the convention for how that is represented, e.g., “unk”, "-999999"

Contoh Kamus Data

- **Concepts** - synonymous with Entity from E/R Modelling
- **Definition of Concepts**
- **Definition in MyHDD and additional source standards** – validate against other standards such as MyHDD, CIHI, WHO, etc whether there exists similar concept
- **Remark** – to help user further understand the concept by having use cases or examples
- **Synonyms** – similar concept definition but different concept name, eg: Concept client is synonymous with Patient.
- **Key attributes** – key information about each concept such as Identification for Person concept.
- **Relationships** – states the relationship between the concepts
- **Table Type** – There are two categories of table type:
    - Master Data concepts (labelled as "Masters") were colored blue in the MyHRDM. Master data is synonymous with reference data
    - Transactional concepts (labelled as "Transactional") were colored pink in the MyHRDM. Transactional Data is data about business events (often system transactions) that have historic significance or were needed for analysis by other systems

MyHRDM metadata is detailed in the MyHRDM Data Dictionary section of this document.

Contoh Kamus Data

Table 3.1: Client Concept

Concept	Client
Definition of Concept	A PERSON who receives health service
Definition in MyHDD	None
Additional source standards for definition	**CIHI** *"A person that has received, is receiving, or is eligible for health care related services or goods."*
Remark	PERSON is a supertype of CLIENT, PROVIDER, NEXT OF KIN, etc
Synonyms	None
Key Attributes	• Identification Documents(ID)
Relationship	Each CLIENT • is a PERSON • may be a DONOR or NEWBORN • may have one or many NEXT OF KIN • may be associated with one or more PLACE • may be the recipient of services and/or goods from one or more PROVIDER • may be a participant in one or more HEALTH SERVICE EVENT • may be associated with more than one client
Table Type(Master or Transaction)	Master

Table 3.18: Newborn Concept

Concept	Newborn
Definition of Concept	A CLIENT aged less than or equal to 28 days of life.
Definition in MyHDD	None
Additional source standards for definition	None
Remark	None
Synonyms	None
Key Attributes	TBD
Relationship	Each NEWBORN • is a CLIENT
Table Type(Master or Transaction)	Master

Kamus data yang ditail

GHO | Visualizations | Indicator Metadata Registry - Google Chrome

apps.who.int/gho/data/node.wrapper.imr?x-id=1

عربي 中文 English Français Русский Español

World Health Organization

Health topics | Data | Media centre | Publications | Countries | Programmes | Governance | About WHO | Search

Global Health Observatory visualizations

- Data analysis
- By theme
- By category
- By indicator
- By country
- Metadata
- About the Observatory
- Search

Indicator Metadata Registry

Choose an indicator

Number of dentistry personnel

Data Type Representation

Count

Topic

Health systems resources

Rationale

The WHO Global Strategy on Human Resources for Health: Workforce 2030 sets out the policy agenda to ensure a workforce that is fit for purpose to attain the targets of the Sustainable Development Goals (SDGs). One of its objectives is primarily linked to strengthening data on human resources for health

Definition

Total number of dentistry personnel in the country.

Associated terms

Classification of health workers : The WHO framework for classifying health workers draws on the latest revisions of international classifications for social and economic statistics, including the International Standard Classification of Occupations (2008 revision), the International Standard Classification of Education (1997

Langkah-langkah

- Susun BPMN → dapatkan service/process, dan informasi yang dimasukkan/dihasilkan pada sistem → entitas pada IDEF 1
- Susun Information modelling IDEF 1 → dapatkan informasi-informasi (melihat pada BPMN)
- Dari entitas pada IDEF 1 → susun data modelling (data yang dibutuhkan utk mengolah entitas pada IDEF 1 → ini menjadi entitas pada IDEF 1x
- Atribute pada entitas IDEF 1x dikumpulkan → susun menjadi Kamus Data
- Tambahkan metadata yang dibutuhkan pada data tersebut

Proses , Info, Data, Kamus Data

Bisnis Proses

Information Modelling

Data Modelling

Kamus Data

Proses/Service

Informasi (IN)

Informasi (OUT)

Informasi → Entitas

Attribute -→Storage Data

Attribute -→Storage Data

Storage Data → Entitas

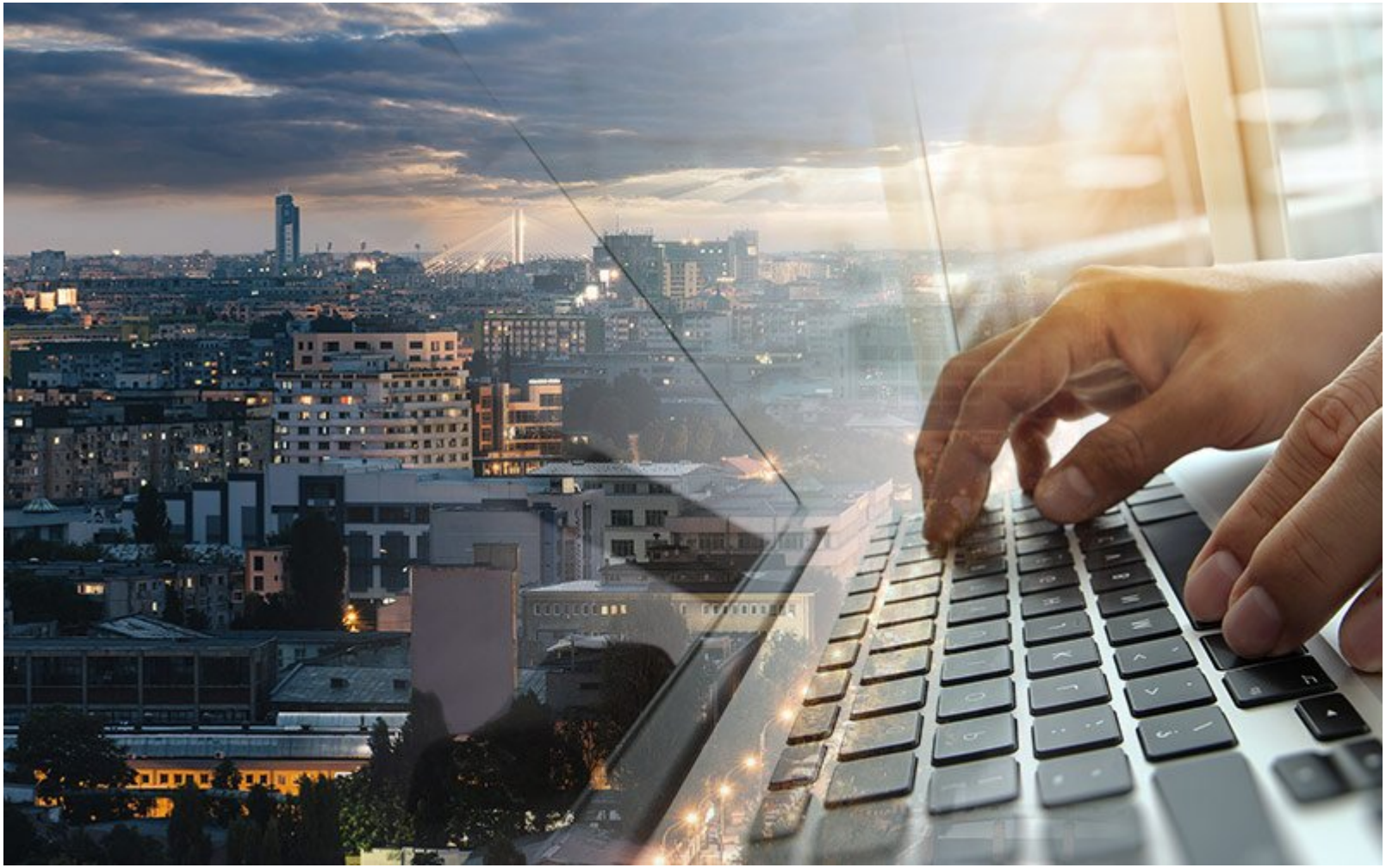
Attribute -->Data

Attribute -->Data

Data

Deskripsi Data

Deskripsi Data



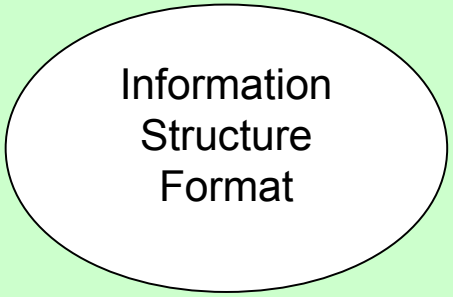
XML Schema

What is XML Schema?

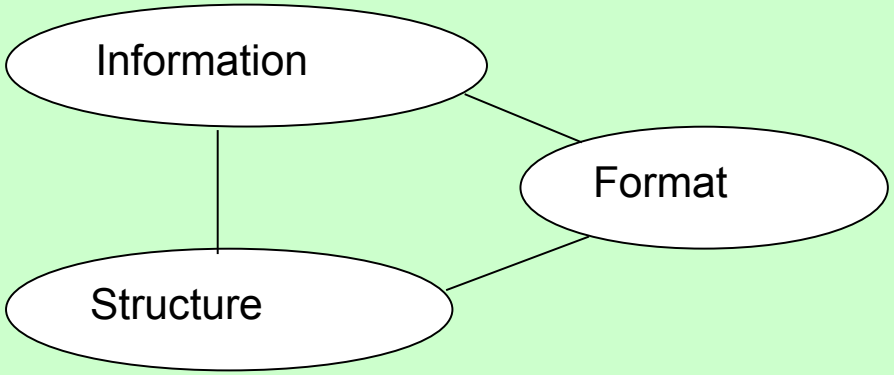
- The origin of schema
  - XML Schema documents are used to define and validate the content and structure of XML data.
  - XML Schema was originally proposed by Microsoft, but became an official W3C recommendation in May 2001

Why Schema? (1)

Separating Information from Structure and Format

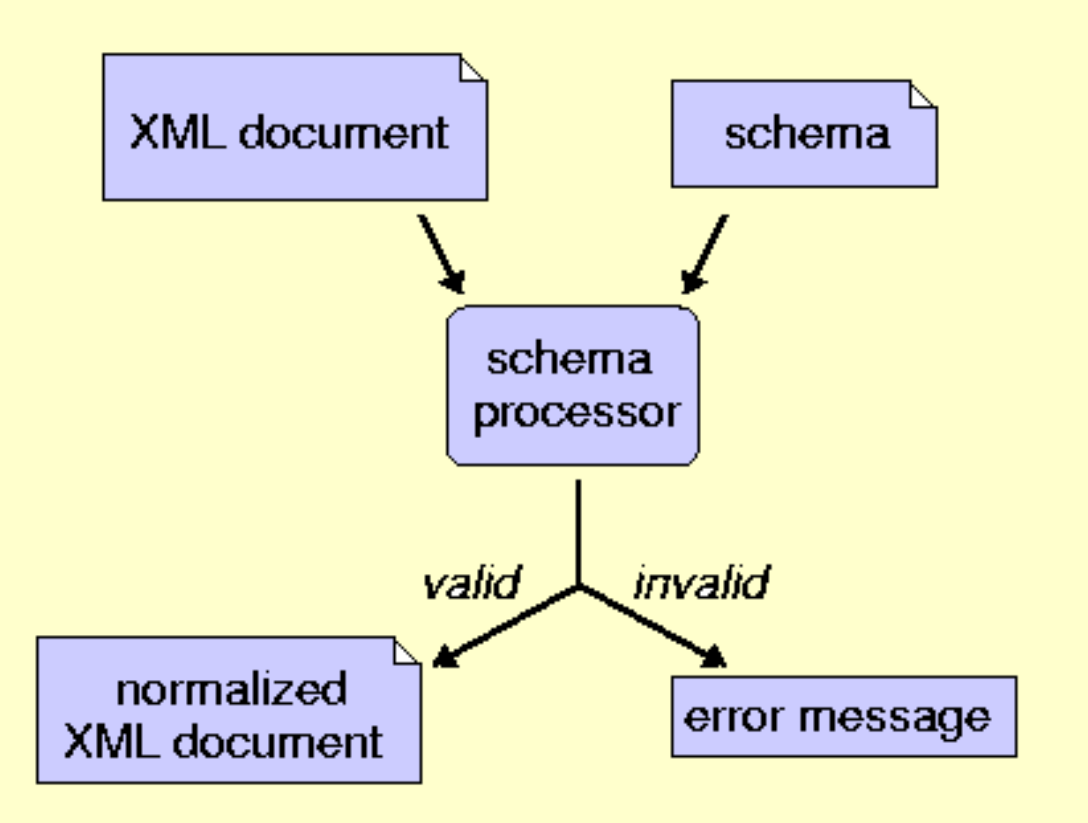


Traditional Document:
Everything is clumped together



"Fashionable" Document: A document is broken into discrete parts, which can be treated separately

Why Schema? (2)



Schema Workflow

DTD versus Schema

Limitations of DTD

- No constraints on character data
- Not using XML syntax
- No support for namespace
- Very limited for reusability and extensibility

Advantages of Schema

- Syntax in XML Style
- Supporting Namespace and import/include
- More data types
- Able to create complex data type by inheritance
- Inheritance by extension or restriction
- More …

Problems of XML Schema

- General Problem
  - Several-hundred-page spec in a very technical language
- Practical Limitations of expressibility
  - content and attribute declarations cannot depend on attributes or element context.
- Technical Problem
  - The notion of “type” adds an extra layer of confusing complexity
- …

XML Schemas

- "Schemas" is a general term--DTDs are a form of XML schemas
  - According to the dictionary, a schema is "a structured framework or plan"
- When we say "XML Schemas," we usually mean the W3C XML Schema Language
  - This is also known as "XML Schema Definition" language, or XSD
  - I'll use "XSD" frequently, because it's short
- DTDs, XML Schemas, and RELAX NG are all XML schema languages

Why XML Schemas?

- DTDs provide a very weak specification language
  - You can't put any restrictions on text content
  - You have very little control over mixed content (text plus elements)
  - You have little control over ordering of elements
- DTDs are written in a strange (non-XML) format
  - You need separate parsers for DTDs and XML
- The XML Schema Definition language solves these problems
  - XSD gives you much more control over structure and content
  - XSD is written in XML

Why not XML schemas?

- DTDs have been around longer than XSD
  - Therefore they are more widely used
  - Also, more tools support them
- XSD is very verbose, even by XML standards
- More advanced XML Schema instructions can be non-intuitive and confusing

- Nevertheless, XSD is not likely to go away quickly

Referring to a schema

- To refer to a DTD in an XML document, the reference goes *before* the root element:
  - <?xml version="1.0"?>
    <!DOCTYPE rootElement SYSTEM "***url***">
    <rootElement> … </rootElement>
- To refer to an XML Schema in an XML document, the reference goes *in* the root element:
  - <?xml version="1.0"?>
    <rootElement
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    (The XML Schema Instance reference is required)
    xsi:noNamespaceSchemaLocation="***url***.xsd">
    (This is where *your* XML Schema definition can be found)
    …
    </rootElement>

The XSD document

- Since the XSD is written in XML, it can get confusing which we are talking about
- Except for the additions to the root element of our XML data document, the rest of this lecture is about the XSD schema document
- The file extension is .xsd
- The root element is <schema>
- The XSD starts like this:
  - <?xml version="1.0"?>
    <xs:schema xmlns:xs="http://www.w3.rg/2001/XMLSchema">

<schema>

- The <schema> element may have attributes:
  - xmlns:xs="http://www.w3.org/2001/XMLSchema"
    - This is necessary to specify where all our XSD tags are defined
  - elementFormDefault="qualified"
    - This means that all XML elements must be qualified (use a namespace)
    - It is highly desirable to qualify all elements, or problems will arise when another schema is added

"Simple" and "complex" elements

- A "simple" element is one that contains text and nothing else
  - A simple element cannot have attributes
  - A simple element cannot contain other elements
  - A simple element cannot be empty
  - However, the text can be of many different types, and may have various restrictions applied to it
- If an element isn't simple, it's "complex"
  - A complex element may have attributes
  - A complex element may be empty, or it may contain text, other elements, or both text and other elements

Defining a simple element

- A simple element is defined as
  <xs:element name="***name***" type="***type***" />
  where:
  - ***name*** is the name of the element
  - the most common values for ***type*** are

xs:boolean	xs:integer
xs:date	xs:string
xs:decimal	xs:time
- Other attributes a simple element may have:
  - default="***default value***" *if no other value is specified*
  - fixed="***value***" *no other value may be specified*

Defining an attribute

- Attributes themselves are always declared as simple types
- An attribute is defined as
  <xs:attribute name="***name***" type="***type***" />
  where:
  - ***name*** and ***type*** are the same as for xs:element
- Other attributes a simple element may have:
  - default="***default value***" *if no other value is specified*
  - fixed="***value***" *no other value may be specified*
  - use="optional" *the attribute is not required (default)*
  - use="required" *the attribute must be present*

Restrictions, or "facets"

- The general form for putting a restriction on a text value is:
  - <xs:element name="*name*"> *(or* xs:attribute*)*

```
<xs:element name="name">
    <xs:restriction base="type">
        … the restrictions …
    </xs:restriction>
</xs:element>
```

- For example:
  - <xs:element name="age">

```
<xs:element name="age">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0">
        <xs:maxInclusive value="140">
    </xs:restriction>
</xs:element>
```

Restrictions on numbers

- minInclusive -- number must be ≥ the given ***value***
- minExclusive -- number must be > the given ***value***
- maxInclusive -- number must be ≤ the given ***value***
- maxExclusive -- number must be < the given ***value***
- totalDigits -- number must have exactly ***value*** digits
- fractionDigits -- number must have no more than ***value*** digits after the decimal point

An XML Instance Document Example

```
<book isbn="0836217462">
    <title> Being a Dog Is a Full-Time Job</title>
    <author>Charles M. Schulz</author>
    <qualification>   extroverted beagle  </qualification>
</book>
```

Restrictions on strings

- length -- the string must contain exactly ***value*** characters
- minLength -- the string must contain at least ***value*** characters
- maxLength -- the string must contain no more than ***value*** characters
- pattern -- the ***value*** is a regular expression that the string must match
- whiteSpace -- not really a “restriction”--tells what to do with whitespace
  - value="preserve" Keep all whitespace
  - value="replace" Change all whitespace characters to spaces
  - value="collapse" Remove leading and trailing whitespace, and replace all sequences of whitespace with a single space

Enumeration

- An enumeration restricts the value to be one of a fixed set of values
- Example:
  -
```
<xs:element name="season">
    <xs:simpleType>
        <xs:restriction  base="xs:string">
            <xs:enumeration value="Spring"/>
            <xs:enumeration value="Summer"/>
            <xs:enumeration value="Autumn"/>
            <xs:enumeration value="Fall"/>
            <xs:enumeration value="Winter"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Complex elements

- A complex element is defined as

```
<xs:element   name="name">
    <xs:complexType>
        ... information about the complex type...
    </xs:complexType>
</xs:element>
```

- Example:

```
<xs:element   name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element  name="firstName"  type="xs:string" />
            <xs:element  name="lastName"  type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- <xs:sequence> says that elements must occur in this order
- Remember that attributes are always simple types

Global and local definitions

- Elements declared at the "top level" of a <schema> are available for use throughout the schema
- Elements declared within a xs:complexType are local to that type
- Thus, in

```
<xs:element   name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element  name="firstName"  type="xs:string" />
            <xs:element  name="lastName"  type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

  the elements firstName and lastName are only locally declared
- The order of declarations at the "top level" of a <schema> *do not* specify the order in the XML data document

Declaration and use

- So far we've been talking about how to *declare* types, not how to *use* them
- To *use* a type we have declared, use it as the value of type="…"
  - Examples:
    - <xs:element name="student" type="person"/>
    - <xs:element name="professor" type="person"/>
  - Scope is important: you cannot use a type if is local to some other type

xs:sequence

- We've already seen an example of a complex type whose elements must occur in a specific order:
-
```
<xs:element   name="person">
   <xs:complexType>
      <xs:sequence>
         <xs:element  name="firstName"  type="xs:string" />
         <xs:element  name="lastName"  type="xs:string" />
      </xs:sequence>
   </xs:complexType>
 </xs:element>
```

xs:all

- xs:all allows elements to appear in any order
-
```
<xs:element   name="person">
   <xs:complexType>
       <xs:all>
           <xs:element  name="firstName"  type="xs:string" />
           <xs:element  name="lastName"  type="xs:string" />
       </xs:all>
   </xs:complexType>
 </xs:element>
```
- Despite the name, the members of an xs:all group can occur once or not at all
- You can use minOccurs="0" to specify that an element is optional (default value is 1)
  - In this context, maxOccurs is always 1

Referencing

- Once you have defined an element or attribute (with name="…"), you can refer to it with ref="…"
- Example:
  - <xs:element name="person">
    <xs:complexType>
    <xs:all>
    <xs:element name="firstName" type="xs:string" />
    <xs:element name="lastName" type="xs:string" />
    </xs:all>
    </xs:complexType>
    </xs:element>
  - <xs:element name="student" ref="person">
  - Or just: <xs:element ref="person">

Text element with attributes

- If a text element has attributes, it is no longer a simple type
  -
```
<xs:element  name="population">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension  base="xs:integer">
                <xs:attribute  name="year”
                               type="xs:integer">
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Empty elements

- Empty elements are (ridiculously) complex

- ```
  <xs:complexType name="counter">
      <xs:complexContent>
          <xs:extension base="xs:anyType"/>
          <xs:attribute name="count" type="xs:integer"/>
      </xs:complexContent>
  </xs:complexType>
  ```
  ```

Mixed elements

- Mixed elements may contain both text and elements
- We add mixed="true" to the xs:complexType element
- The text itself is not mentioned in the element, and may go anywhere (it is basically ignored)

-
```
<xs:complexType  name="paragraph"  mixed="true">
    <xs:sequence>
        <xs:element  name="someName”
                     type="xs:anyType"/>
    </xs:sequence>
</xs:complexType>
```

Extensions

- You can base a complex type on another complex type
-
```
<xs:complexType  name="newType">
    <xs:complexContent>
        <xs:extension  base="otherType">
            ...new stuff...
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Predefined string types

- Recall that a simple element is defined as:
  <xs:element name="*name*" type="*type*" />
- Here are a few of the possible string types:
  - xs:string -- a string
  - xs:normalizedString -- a string that doesn't contain tabs, newlines, or carriage returns
  - xs:token -- a string that doesn't contain any whitespace other than single spaces
- Allowable restrictions on strings:
  - enumeration, length, maxLength, minLength, pattern, whiteSpace

Predefined date and time types

- xs:date -- A date in the format ***CCYY-MM-DD***, for example, 2002-11-05
- xs:time -- A date in the format ***hh:mm:ss*** (hours, minutes, seconds)
- xs:dateTime -- Format is ***CCYY-MM-DD*T*hh:mm:ss***
  - The T is part of the syntax
- Allowable restrictions on dates and times:
  - enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, pattern, whiteSpace

Predefined numeric types

- Here are some of the predefined numeric types:

xs:decimal	xs:positiveInteger
xs:byte	xs:negativeInteger
xs:short	xs:nonPositiveInteger
xs:int	xs:nonNegativeInteger
xs:long	

- Allowable restrictions on numeric types:
  - enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, fractionDigits, totalDigits, pattern, whiteSpace

The Example's Schema

```
<?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="book">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="title" type="xs:string"/>
               <xs:element name="author" type="xs:string"/>
               <xs:element name="qualification" type="xs:string"/>
             </xs:sequence>
            </xs:complexType>
          </xs:element>
</xs:schema>
```

book.xsd

XML Abstract Data Model

- The XML Abstract Data Model
  - composes of Schema Components.
  - is used to describe XML Schemas.
- Schema Component
  - is the generic term for the building blocks that compose the abstract data model of the schema.

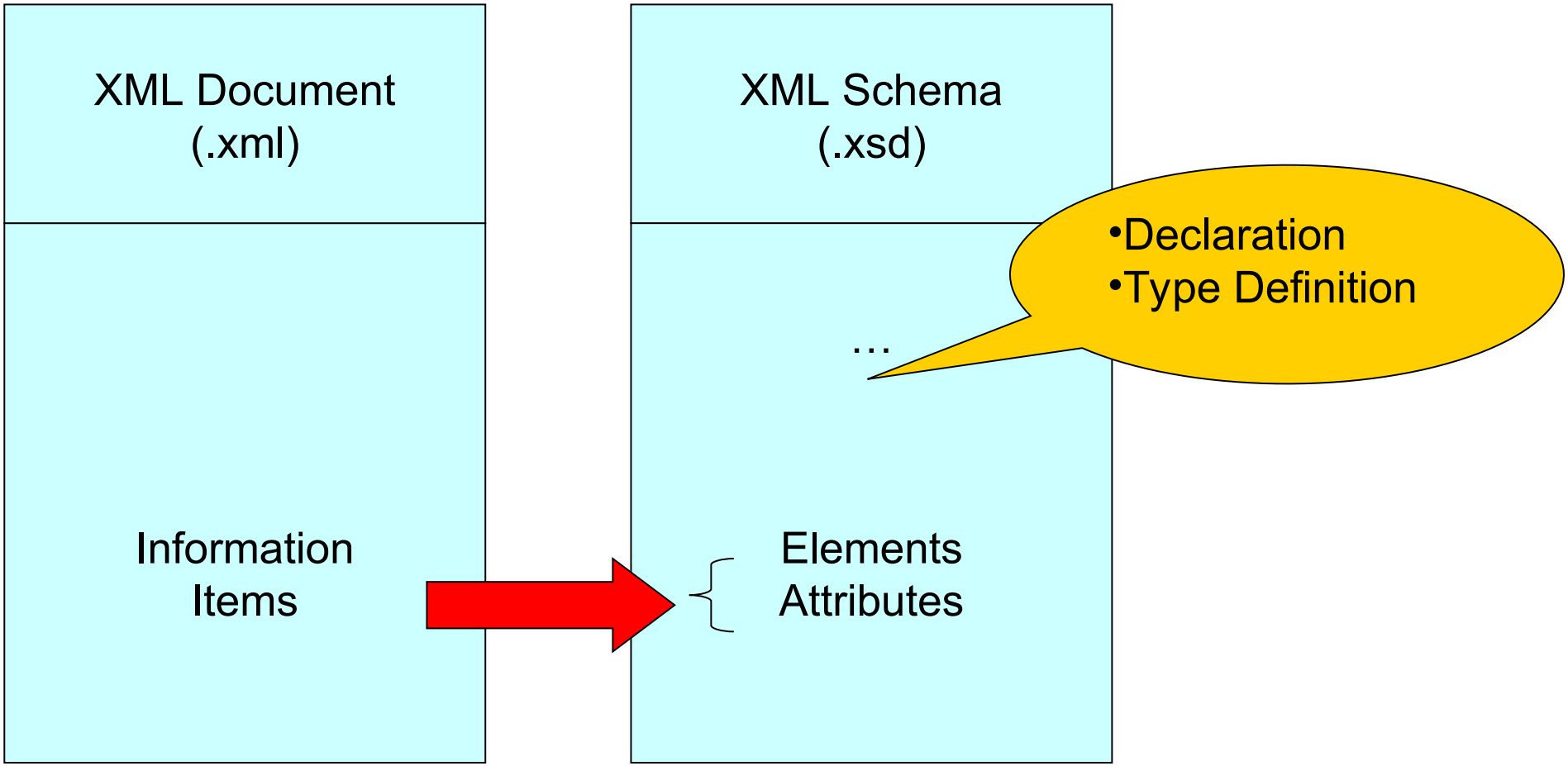
13 Kinds of Schema Components

- Simple type definitions
- Complex type definitions
- Attribute declarations
- Element declarations

- Attribute group definitions
- Identity-constraint definitions
- Model group definitions
- Notation declarations

- Annotations
- Model groups
- Particles
- Wildcards
- Attribute Uses

XML document & XML Schema



Declaration & Definition

- Declaration Components
  - are associated by (qualified) names to information items being validated.
  - It is like declaring objects in OOP.
- Definition Components
  - define internal schema components that can be used in other schema components.
  - Type definition is like defining classes in OOP.

```
<book isbn="0836217462">
 <title>
  Being a Dog Is a Full-Time Job
 </title>
.....
</book>
```

Declaration

```
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
            ... ...
    </xs:sequence>
    <xs:attribute name="isbn" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

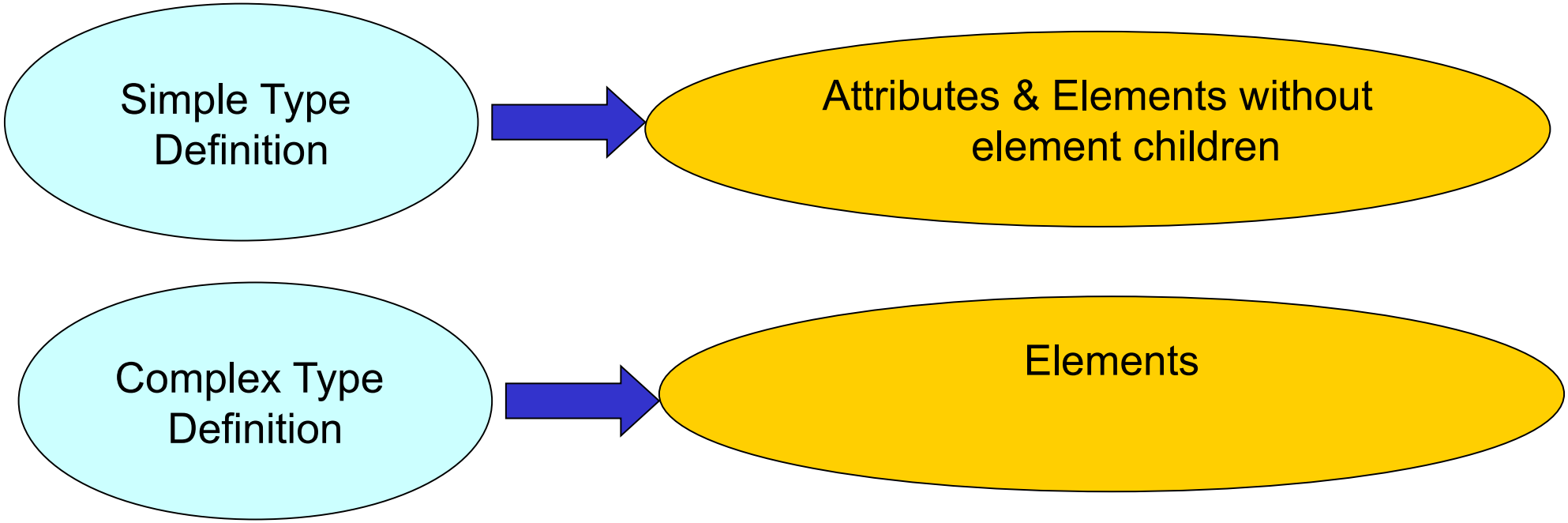
```
<xs:element name="book" type="bookType"/>
```

Type Definition

```
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:element name="title" type="nameType"/>
      ... ...
  </xs:sequence>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```

Type Definitions

- Why Type Definitions?
- Simple Type Definition VS. Complex Type Definition



Simple Type Definition

- Simple Type Definition can be:
  - a restriction of some other simple type;
  - a list or union of simple type definition;or
  - a built-in primitive datatypes.
- Example

```
<xs:simpleType name="farenheitWaterTemp">
 <xs:restriction base="xs:number">
  <xs:fractionDigits value="2"/>
  <xs:minExclusive value="0.00"/>
  <xs:maxExclusive value="100.00"/>
 </xs:restriction>
</xs:simpleType>
```

Complex Type Definition(1)

- Inheritance
  - Restriction: We restrict base types definitions.
  - Extension: We add something new.
- Composition
  - Group model

Complex Type Definition(2)

- Inheritance

  Each complex type definition is either
  - a restriction of a complex type definition
  - an extension of a simple or complex type definition
  - a restriction of the ur-type definition.
- Example

```
<xs:complexType name="personName">
 <xs:sequence>
  <xs:element name="title" minOccurs="0"/>
      ... ...
 </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="extendedName">
 <xs:complexContent>
  <xs:extension base="personName">
   <xs:sequence>
    <xs:element name="generation"
                minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Complex Type Definition(3)

- Composition

  Model Group is composed of

  - Compositor (*sequence* |*choice*| *all*)
  - Particles, which can be
    - Element Declaration
    - Wildcard
    - Model Group

Complex Type Definition(4)

- Model Group Examples

```
<xs:all>
 <xs:element ref="cats"/>
 <xs:element ref="dogs"/>
</xs:all>
```

```
<xs:sequence>
 <xs:choice>
  <xs:element ref="left"/>
  <xs:element ref="right"/>
 </xs:choice>
 <xs:element ref="landmark"/>
</xs:sequence>
```

Complex Type Definition(5)

- Reusable Fragments of Type Definition
  - Model group definition
  - Attribute group definition

```
<xs:group name="mainBookElements">
  <xs:sequence>
    <xs:element name="title" type="nameType"/>
    <xs:element name="author" type="nameType"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="bookType">
  <xs:sequence>
    <xs:group ref="mainBookElements"/>
      ... ...
  </xs:sequence>
  <xs:attributeGroup ref="bookAttributes"/>
</xs:complexType>
```

Declarations(1)

- Element Declaration
- Attribute Declaration
  - Attribute uses
- Notation Declaration
  - Notation declarations reconstruct XML 1.0 NOTATION declarations.

Declarations(2)

Examples

```
<xs:element name="PurchaseOrder" type="PurchaseOrderType"/>
```

```
<xs:element name="gift">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="birthday" type="xs:date"/>
   <xs:element ref="PurchaseOrder"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

```
<xs:attribute name="age" type="xs:positiveInteger" use="required"/>
```

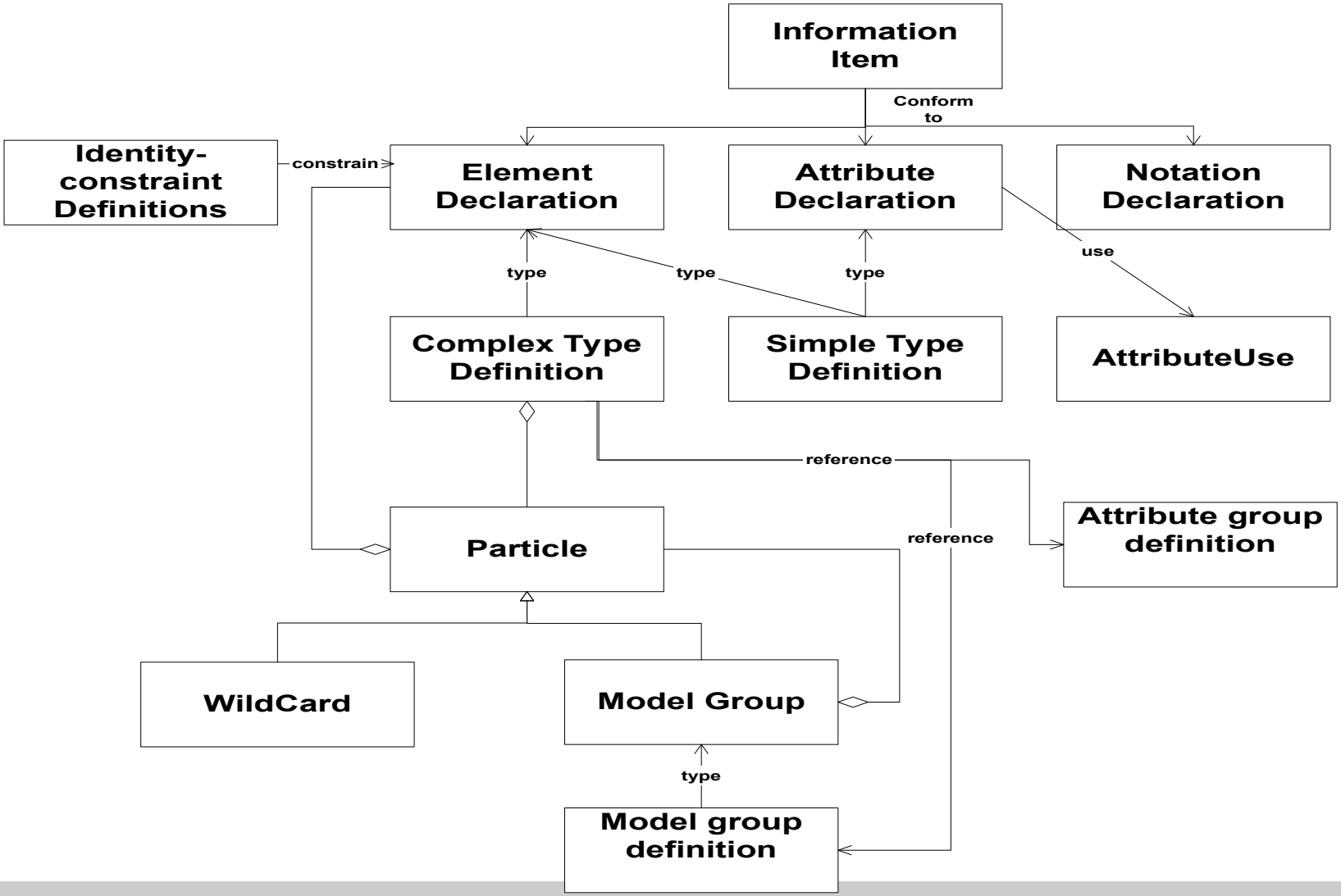
```
 <xs:notation name="jpeg" public="image/jpeg" system="viewer.exe">
```

Annotations

- Annotations provide for human- and machine-targeted documentations of schema components.
- They are meta-data.

```
<xs:simpleType fn:note="special">
 <xs:annotation>
  <xs:documentation>A type for experts only</xs:documentation>
  <xs:appinfo>
   <fn:specialHandling>checkForPrimes</fn:specialHandling>
  </xs:appinfo>
 </xs:annotation>
```

Relationships among Schema Components



XML Schema Requirements

- Structural
  - namespaces
  - primitive types & structural schema integration
  - inheritance
- Data type
  - integers, dates, … (like in languages)
  - user-defined (constrain some properties)
- Conformance
  - processors, validity

Design Principles

- More expressive than DTDs
- Expressed in XML
- Self-describing
- Usable by various XML applications
- Simple enough

The Specifications

- Part 0: Primer
  - non-normative introduction
- Part 1: Structures
  - define structure
  - constraining contents
- Part 2: Datatypes
  - specify datatypes on elements and attributes

An Example Document (1/2)

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
 <shipTo country="US">
  <name>Matthias Hauswirth</name>
  <street>4500 Brookfield Dr.</street>
  <city>Boulder</city>
  <state>CO</state>
  <zip>80303</zip>
 </shipTo>
 <billTo country="US">
  <name>Brian Temple</name>
  <street>1234 Strasse</street>
  <city>Boulder</city>
  <state>CO</state>
  <zip>80302</zip>
 </billTo>
 ...
```

An Example Document (2/2)

```
 <comment>Brian pays</comment>
 <items>
  <item partNum="123-AB">
   <productName>Porsche</productName>
   <quantity>1</quantity>
   <price>129400.00</price>
   <comment>Need a new one</comment>
  </item>
  <item>
   <productName>Ferrari</productName>
   <quantity>2</quantity>
   <price>189000.25</price>
   <shipDate>1999-05-21</shipDate>
  </item>
 </items>
</purchaseOrder>
```

An Example Schema (1/3)

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

   <xsd:element name="purchaseOrder" type="purchaseOrderType"/>

   <xsd:element name="comment" type="xsd:string"/>

   <xsd:complexType name="PurchaseOrderType">
        <xsd:element name="shipTo" type="AddressType"/>
        <xsd:element name="billTo" type="AddressType"/>
        <xsd:element ref="comment" minOccurs="0"/>
        <xsd:element name="items" type="ItemsType"/>
        <xsd:attribute name="orderDate" type="xsd:date"/>
   </xsd:complexType>

   ...
```

An Example Schema (2/3)

```
<xsd:complexType name="AddressType">
     <xsd:element name="name" type="xsd:string/>
     <xsd:element name="street" type="xsd:string"/>
     <xsd:element name="city" type="xsd:string"/>
     <xsd:element name="state" type="xsd:string"/>
     <xsd:element name="zip" type="xsd:decimal"/>
     <xsd:attribute name="country" type="xsd:NMTOKEN"
            use="fixed" value="US"/>
</xsd:complexType>

<xsd:simpleType name="SkuType" base="xsd:string">
     <xsd:pattern value="\d{3}-[A-Z]{2}"/>
</xsd:simpleType>

...
```

An Example Schema (3/3)

```
<xsd:complexType name="ItemsType">
 <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
   <xsd:element name="productName" type="xsd:string/>
   <xsd:element name="quantity">
    <xsd:simpleType base="xsd:positiveInteger">
     <xsd:maxExclusive Value="100"/>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="price" type="xsd:decimal"/>
   <xsd:element ref="comment" minOccurs="0"/>
   <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
   <xsd:attribute name="partNum" type="SkuType"/>
  </xsd:complexType>
 </xsd:element>
</xsd:complexType>
</xsd:schema>
```

Part 1: Structures

- Type Definitions ***<simpleType>*** **<complexType>**
  `<element> <group> <all> <choice> <sequence>`
  `<attribute> <attributeGroup>`
- Attribute Declarations **<attribute>**
  <simpleType>
- Element Declarations **<element>**
  <simpleType> `<complexType>`
- Attribute Group Definitions **<attributeGroup>**
  `<attribute> <attributeGroup>`
- Model Group Definitions **<group>**
  `<element> <group> <all> <choice> <sequence>`
- Notation Declarations **<notation>**
- Annotations **<annotation>**
  `<appinfo> <documentation>`

DTD vs. Schema Structure

- DTD

```
<!ELEMENT e1
   ((e2,e3?)+|e4)>
```

- Schema

```
<element name="e1">
 <complexType>
  <choice>
   <sequence maxOccurs="unbounded">
    <element ref="e2"/>
    <element ref="e3" minOccurs="0"/>
   </sequence>
   <element ref="e4">
  </choice>
 </complexType>
</element>
```

Referential/Uniqueness Integrity

- *Define Constraints using XPath expressions*
- `<unique>`
- `<key>`
- `<keyref>`
- `<selector>`
- `<field>`

Part 2: Datatypes `<simpleType>`

- Value Space
  - defined axiomatically (primitive types)
  - enumerated outright
  - defined by restricting value space of other type
  - combination of values of other type (list)
  - *has certain properties (e.g. cardinality, equality, ordered)*
- Lexical Space
  - set of literals for a type (e.g. 100 and 1.0E2 denote same value)
- Facets
  - fundamental facets (define the type)
  - constraining facets (allow to constrain the value space)

Fundamental Facets

- *Fundamental facets can't be changed*
- Equal
  - all types provide an equality relation
- Order
  - some types provide an ordering relation
- Bounds
  - upper bound and lower bound
- Cardinality
  - finite, infinite
- Numeric
  - yes or no

Constraining Facets

- length
- minLength
- maxLength
- pattern
- enumeration
- maxInclusive / maxExclusive
- minInclusive / minExclusive
- precision
- scale
- encoding
- duration
- period

Primitive vs. Derived Types

- Primitive Types
  - string
  - boolean
  - float
  - double
  - decimal
  - timeDuration
  - recurringDuration
  - binary
  - uriReference
  - ID
  - IDREF
  - ENTITY
  - NOTATION
  - QName
- *exist ab initio*

- Derived Type
  - by restriction
    - use constraining facets

```
<simpleType name="sku"
   base="xsd:string>
 <pattern
   value="\d{3}-[A-D]{4}"/>
</simpleType>
```

  - by list
    - next slide

Built-in vs. User-Derived Types

- Built-in types
  - primitive
  - derived
    - language
    - IDREFS
    - long
    - int
    - short
    - positiveInteger
    - time
    - month
    - recurringDay
    - ...
- User-derived types
  - derived-only

Atomic vs. List Types

- Atomic
  - values indivisible

```
<simpleType name="ShoeSize"
    base="xsd:decimal"/>

<element name="shoe"
    type="ShoeSize"/>

<shoe>10.5</shoe>
```

- List
  - sequence of values of atomic type

```
<simpleType name="ShoeSizes"
    base="shoeSize"
    derivedBy="list"/>

<element name="shoes"
    type="ShoeSizes"/>

<shoes>8 10 10.5</shoes>
```

Tools

- XML Schema-aware Parser
  - Xerces-J
  - Oracle XML Schema Processor
- XML Schema Validator (XSV, online)
- DTD to Schema Conversion Tools
- XML Schema Editor
  - Extensibility's XML Authority
- XML Schema-aware Instance Editor
  - Extensibility's XMLInstance
  - ChannelPoint's Merlot (maybe in future)

XML

```
<bibliography>
    <book>   <title> Foundations… </title>
            <author> Abiteboul </author>
            <author> Hull </author>
            <author> Vianu </author>
            <publisher> Addison Wesley </publisher>
            <year> 1995 </year>
    </book>
    …
</bibliography>
```

XML describes the content

XML Terminology

- tags: book, title, author, …
- start tag: <book>, end tag: </book>
- elements: <book>…<book>,<author>…</author>
- elements are nested
- empty element: <red></red> abbrv. <red/>
- an XML document: single *root element*

well formed XML document: if it has matching tags

XML: Attributes

```
<book price = "55" currency = "USD">
  <title> Foundations of Databases </title>
  <author> Abiteboul </author>
  ...
  <year> 1995 </year>
</book>
```

attributes are alternative ways to represent data

Schemas for XML

DTD - Element Type Definitions

```
<!ELEMENT paper (title,author*, year, (journal|conference) )>
```

XML Schemas

- generalizes DTDs (SGML derivative)
- now, instead uses XML syntax
- two main documents: structure and data types

- XML Schema more powerful but more complex

XML Schema

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
   <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" minOccurs="0"/>
      <xsd:element name="year"/>
      <xsd: choice> < xsd:element name="journal"/>
                  <xsd:element name="conference"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType
</xsd:element>
```

DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

So Far

- Differences between “xml schema” versus “relational schema” ?
  - Purpose ? Do we need it ?
  - Definition time?
  - Strictness of typing ?
  - Underlying model ?

Elements versus Types in XML Schema

DTD: <!ELEMENT person (name, address) >

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name"
                   type="xsd:string"/>
      <xsd:element name="address"
                   type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="person"
             type="ttt" />
<xsd:complexType name="ttt">
  <xsd:sequence>
    <xsd:element name="name"
                 type="xsd:string"/>
    <xsd:element name="address"
                 type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Elements versus Types in XML Schema

- Types:
  - Simple types (integers, strings, ...)
  - Complex types (regular expressions, like in DTDs)

- Element-type-element alternation:
  - Root element has a complex type
  - Complex type is a regular expression of elements
  - Those elements have their complex types ...
  - ...
  - Leaves have simple types

Local and Global Types in XML Schema

- Local type:

```
<xsd:element name="person">
        [define locally the person's type]
 </xsd:element>
```

- Global type:

```
<xsd:element name="person" type="ttt"/>

<xsd:complexType name="ttt">
        [define here the type ttt]
</xsd:complexType>
```

Global types: can be reused in other elements

Local v.s. Global Elements in XML Schema

- Local element:

```
<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element name="address"  type="..."/>...
    </xsd:sequence>
  </xsd:complexType>
```

- Global element:

```
<xsd:element name="address" type="..."/>

<xsd:complexType name="ttt">
    <xsd:sequence>
     <xsd:element ref="address"/>  ...
   </xsd:sequence>
 </xsd:complexType>
```

Global elements: like in DTDs

Keys in XML Schema

XML:

```
<purchaseReport>
    <regions>
     <zip code="95819">
            <part number="872-AA" quantity="1"/>
            <part number="926-AA" quantity="1"/>
            <part number="833-AA" quantity="1"/>
            <part number="455-BX" quantity="1"/>
     </zip>
     <zip code="63143">
            <part number="455-BX" quantity="4"/>
     </zip>
    </regions>

    <parts>
            <part number="872-AA">Lawnmower</part>
            <part number="926-AA">Baby Monitor</part>
            <part number="833-AA">Lapis Necklace</part>
            <part number="455-BX">Sturdy Shelves</part>
    </parts>
</purchaseReport>
```

XML Schema for Key :

```
<key name="NumKey">
  <selector xpath="parts/part"/>
  <field xpath="@number"/>
</key>
```

Keys in XML Schema

- In general, syntax is :

```
<key name="someDummyNameHere">
  <selector xpath="p"/>
  <field xpath="p1"/>
  <field xpath="p2"/>
  . . .
  <field xpath="pk"/>
</key>
```

Notes:
All XPath expressions "start" at the element currently being defined
The fields must identify a single "node".

Keys in XML Schema

- Unique = guarantees uniqueness
- Key = guarantees uniqueness and existence
- All XPath expressions are "restricted":
  - /a/b | /a/c OK for selector
  - //a/b/*/c OK for field

- Note: better than DTD's ID mechanism

Examples of Keys in XML Schema

- Examples

Note: Must have single firstname, Single surname

```
<key name="fullName">
    <selector xpath=".//person"/>
    <field xpath="firstname"/>
    <field xpath="surname"/>
</key>

<unique name="nearlyID">
    <selector xpath=".//*"/>
    <field xpath="@id"/>
</unique>
```

Foreign Keys in XML Schema

- Example

```
<keyref name="personRef" refer="fullName">
    <selector xpath=".//personPointer"/>
    <field xpath="@first"/>
    <field xpath="@last"/>
</keyref>
```

Sejarah dan Aktifitas

- 1962, komputer pertama diinstal di Indonesia (PT Stanvac)
- 1965, kelompok pengguna IBM
- 20 Mei 1965, Badan Kerjasama Otomasi Negara (BAKOTAN)
- 18 April 1974 - Himpunan Pengguna Komputer Indonesia (HPKI)
- 20 Juli 1974 – Ikatan Pengguna Komputer Indonesia (IPKIN)
- 1975 IPKIN berbadan hukum
- 1990, Ikatan Profesi Komputer dan Informatika Indonesia

KODE ETIK

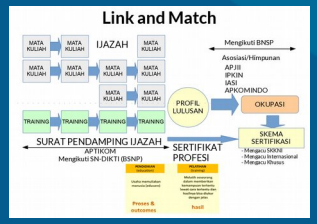
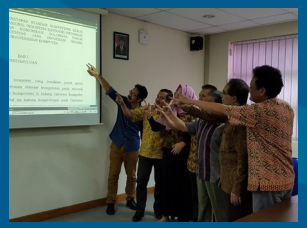
Prinsip Standar Teknis
Prinsip Kompetensi
Prinsip Tanggung Jawab Profesi
Prinsip Kepentingan Publik
Prinsip Integritas
Prinsip Objektivitas
Prinsip Kerahasiaan
Prinsip Perilaku Profesional

- 1979 : Konferensi Komputer Nasiponal (KKN) Pertama di Indonesia,
- 1980: Konferensi SEARCC 3 di Indonesia
- Kode Etik Profesi bidang TIK
- Doktoral dan Master Kollokuium, Klinik Jurnal Ilmiah (Dirjen Dikti – HIVOS), Internasional Conference dan Journal Ilmiah
- Pemetaan dan survey SDM TIK (Kominfo, Aptikom)
- Pengembangan Standard Profesi dari era SRIG-PS 1995 dengan SEARCC, Pengembangan Standard Kompetensi Kerja Nasional Inodnesia (SKKNI, Pemaketan Kerangka Kualifikasi Nasional Indonesia (KKNI) dan Peta Okupasi (dari tahun 2005) – Kominfo, Kemnaker,

IPKIN

Ikatan Profesi Komputer dan Informatika Indonesia
http://www.ipkin.or.id

Perjalanan Keprofesian TIK

The first computer in Indonesia

SEARCC chief delegates at the head table