BACS3183
Advanced Database Management

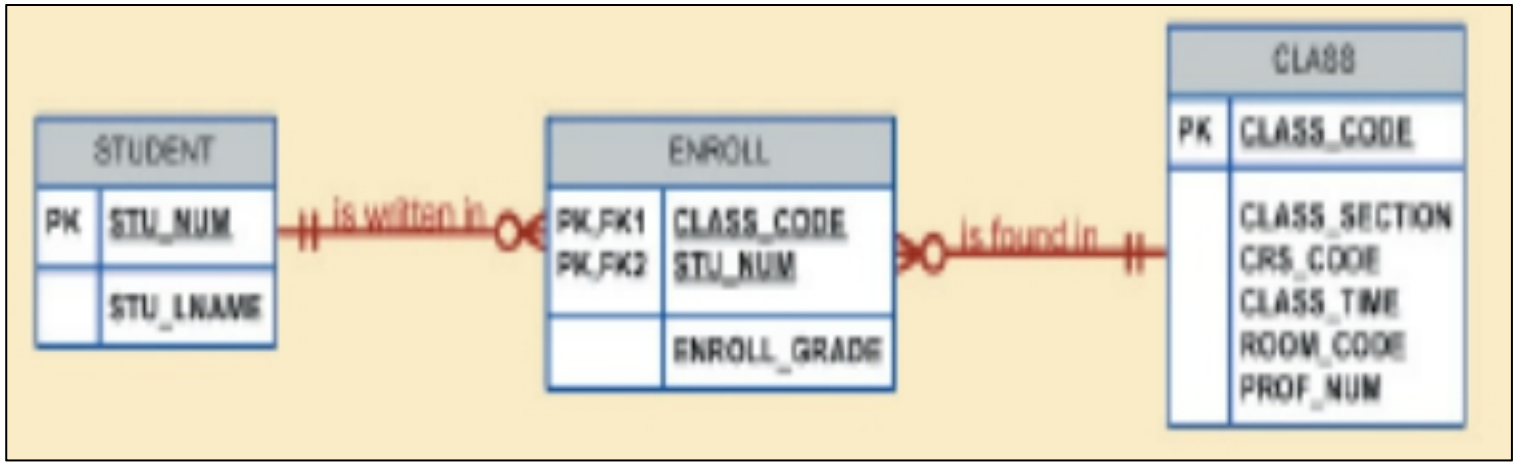
Chapter 3
Data Modeling

Learning Outcomes

- Compare and contrast appropriate data models.
- Describe concepts in conceptual modeling notation.
- Describe the main concepts of the relational, object-oriented and semi-structured data models.

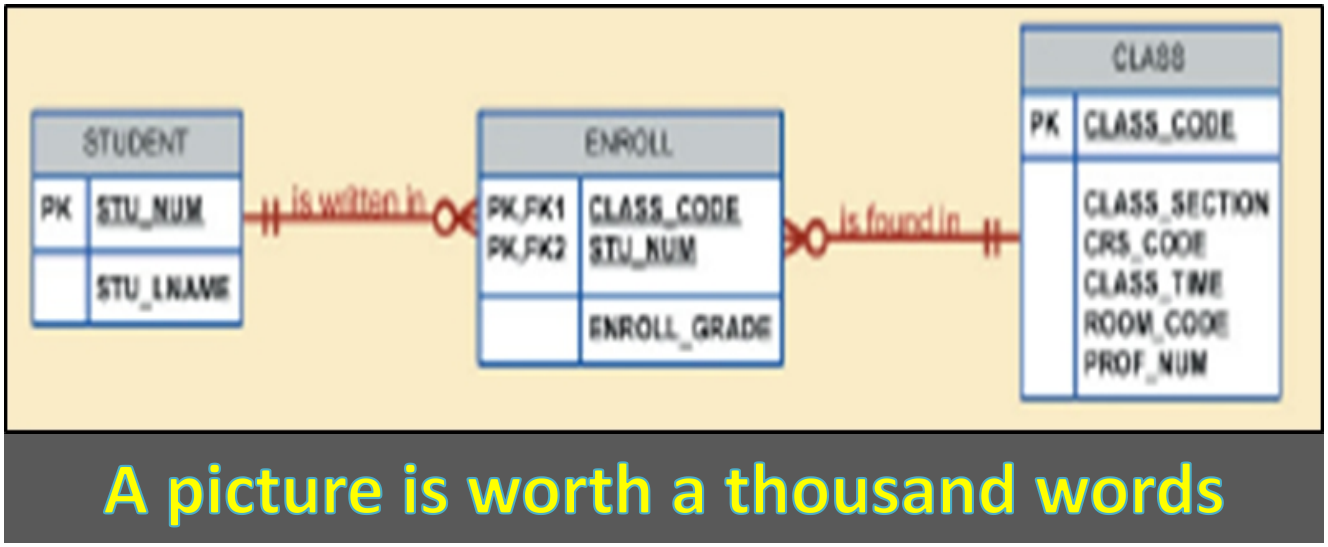
1. Data Modeling

- **Data Modeling**
  - **the first step in designing a database**
  - **the process of creating a specific data model for an information system.**
  - **an iterative and progressive process**



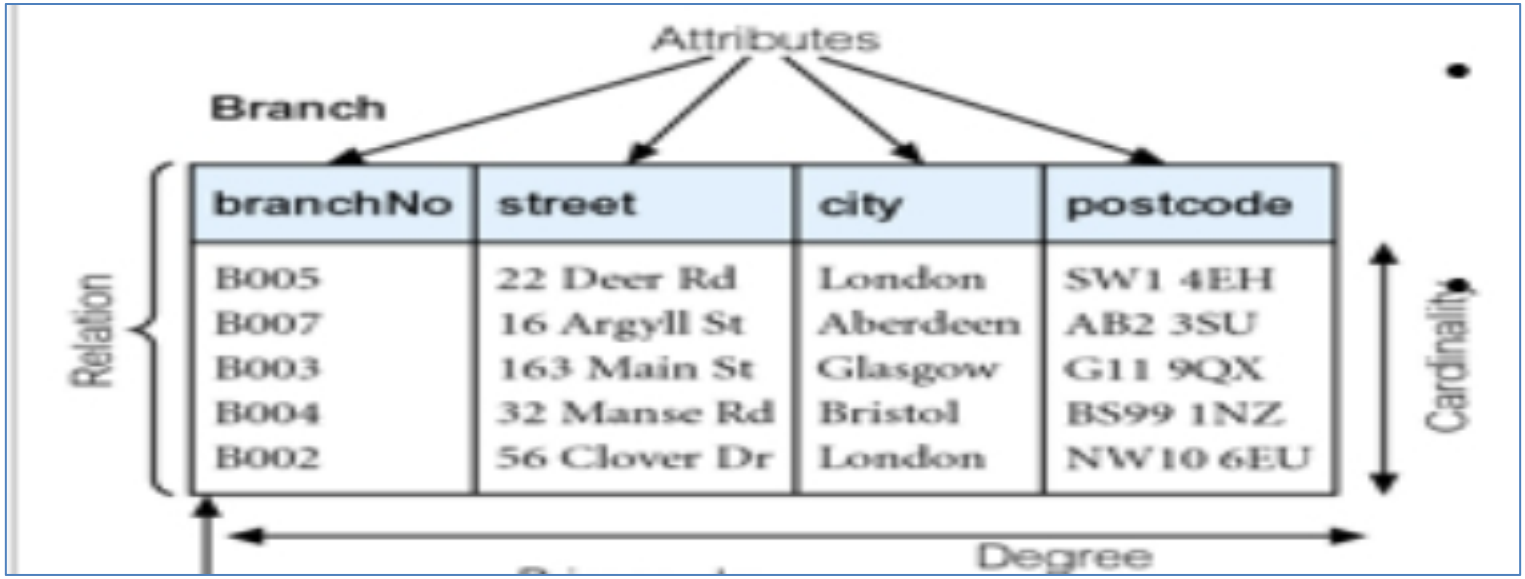
Importance of Data Models

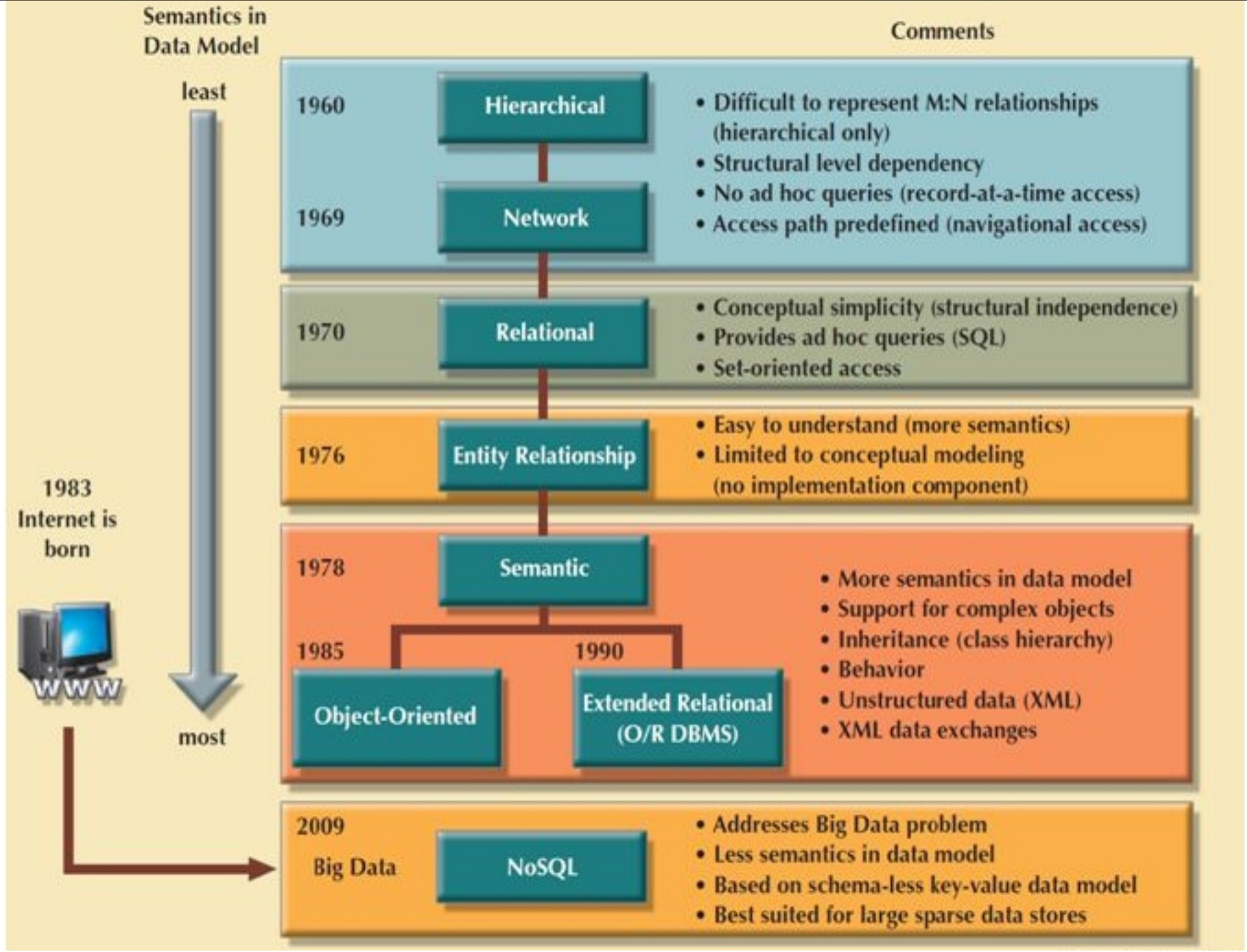
- Data models can facilitate interaction among the designer, the applications programmer, and the end user.
- A well-developed data model can foster improved understanding of the organization for which the database design is developed.
  - Give an overall view of the database
  - Organize data for various users
  - Are an abstraction for the creation of good database



Data Modeling

- **Data model is a collection of concepts for describing**
  - **a structural part**
  - **a manipulative part** - types of operation that are allowed on the data
  - **possibly a set of integrity rules** which ensures that the data is accurate.

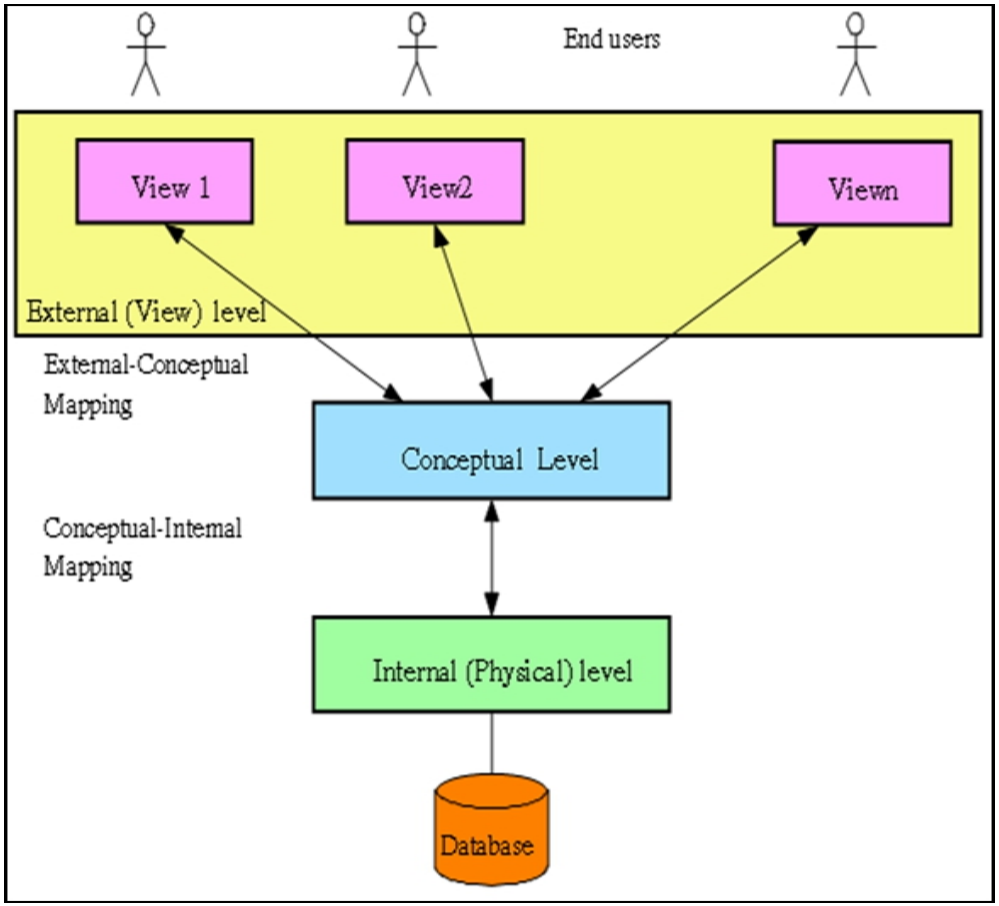


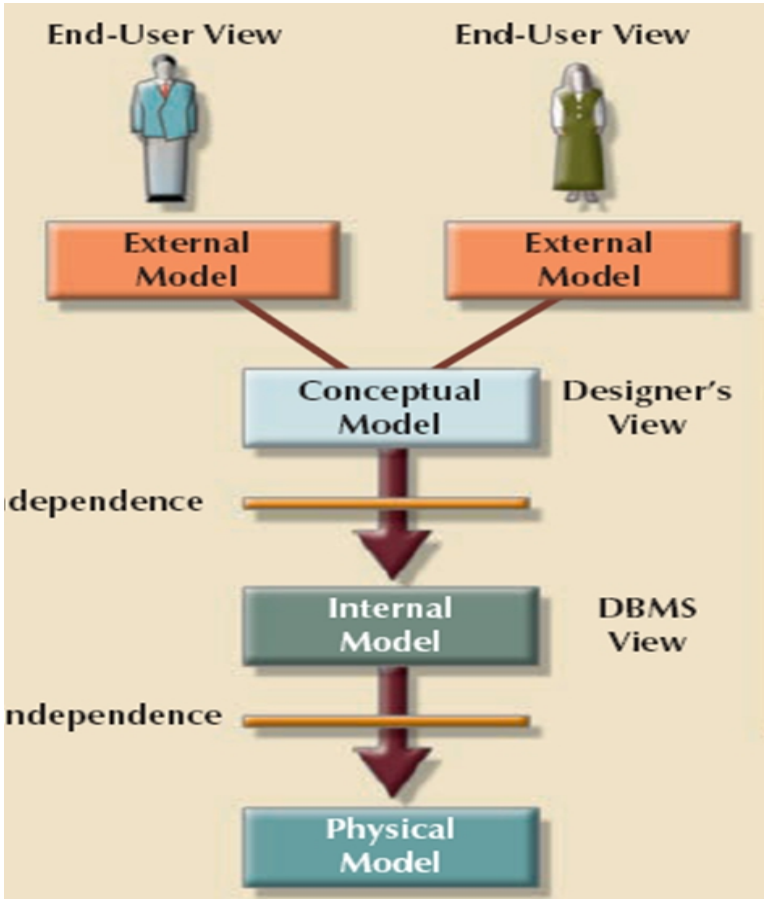


Database system: Design, Implementation & Management, Rob & Colonel

Data Modeling

ANSI-SPARC (*American National Standards Institute, Standards Planning And Requirements Committee)* **defined a framework for data modeling based on degrees of data abstraction**

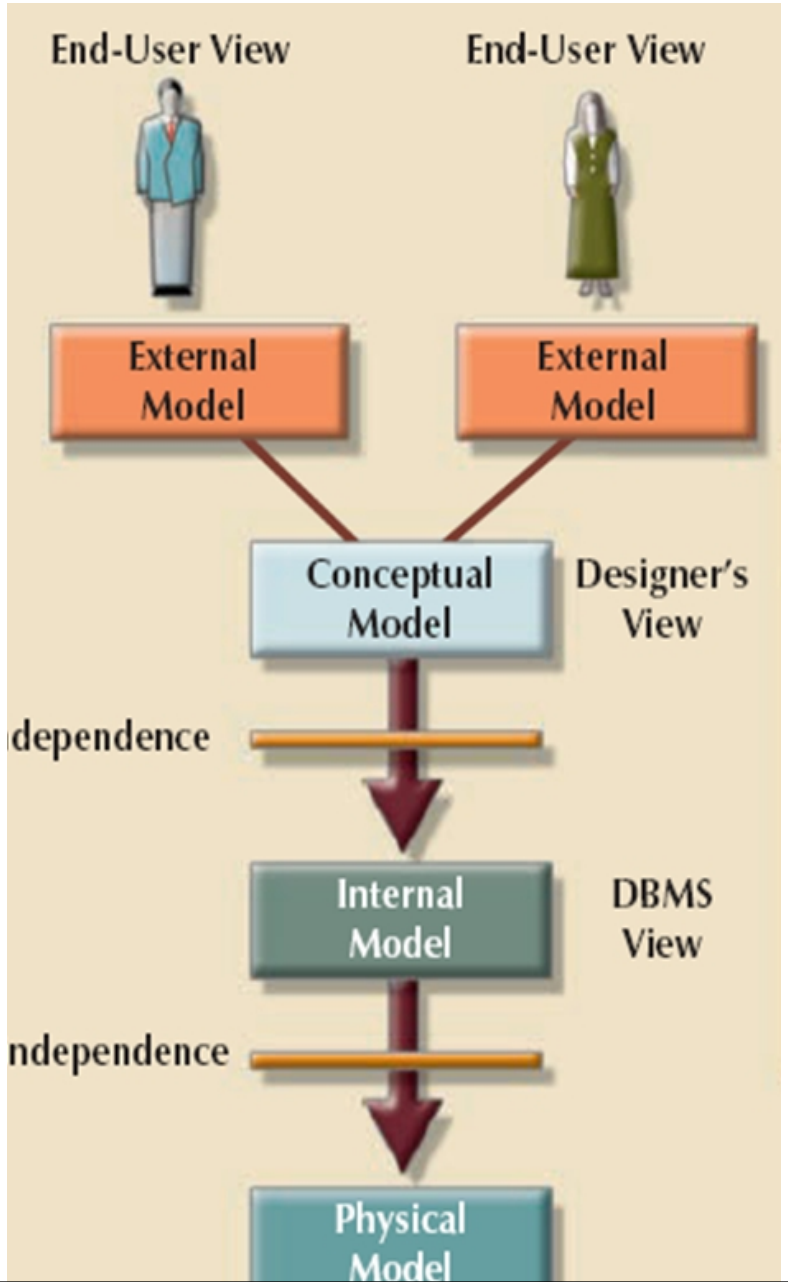




Most modern commercial DBMS are based on this framework.

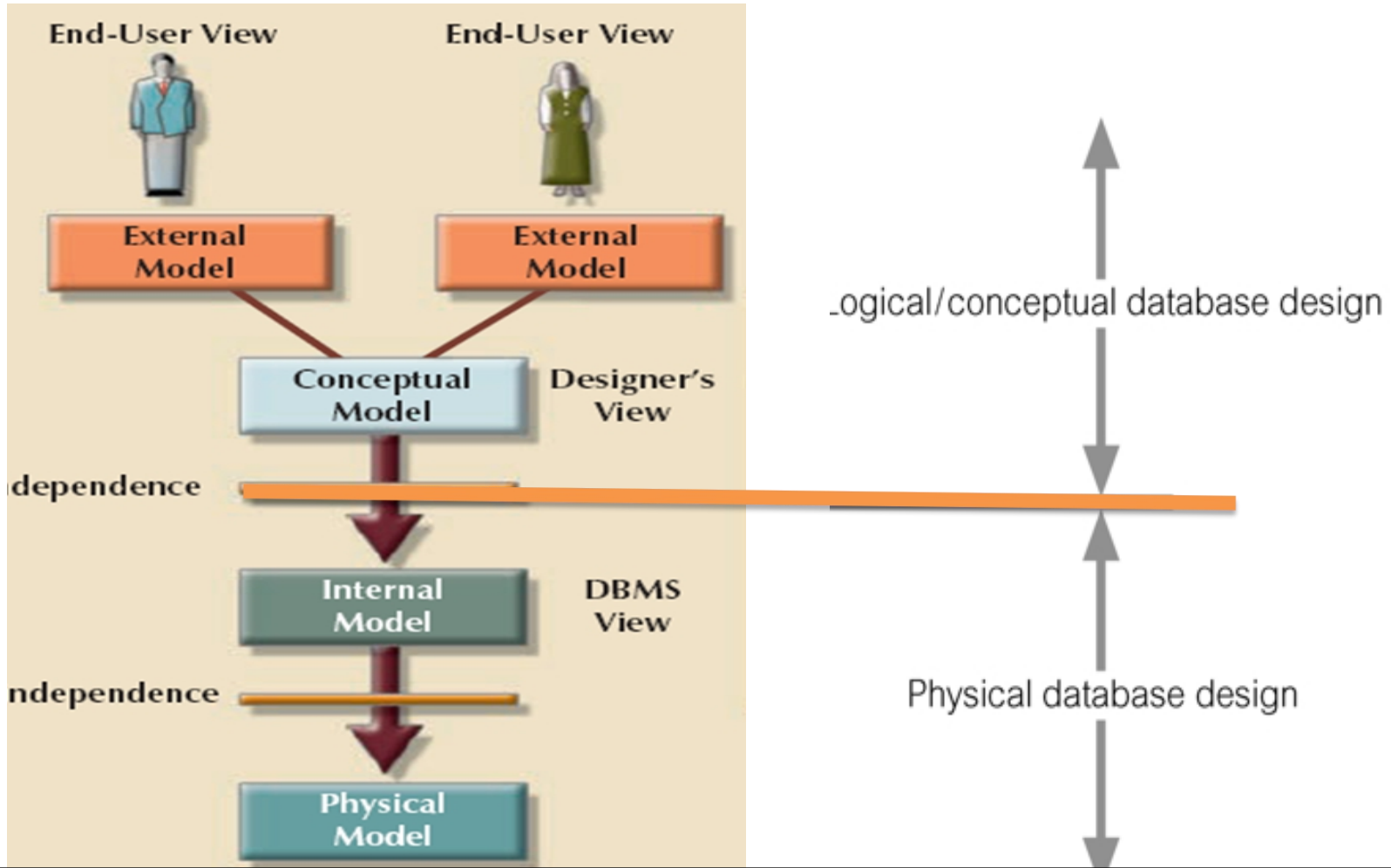
Data Modeling

- **External model:**
  - **End-user view** of the data environment
  - Can be many different models
- **Conceptual model**
  - Describes ***what*** data to be stored used in an **enterprise**, independent of *all* physical considerations.
  - Describes the **relationships** among data.
- **Internal model**
  - **How** the data are stored.
  - Complex low-level structures described in detail.



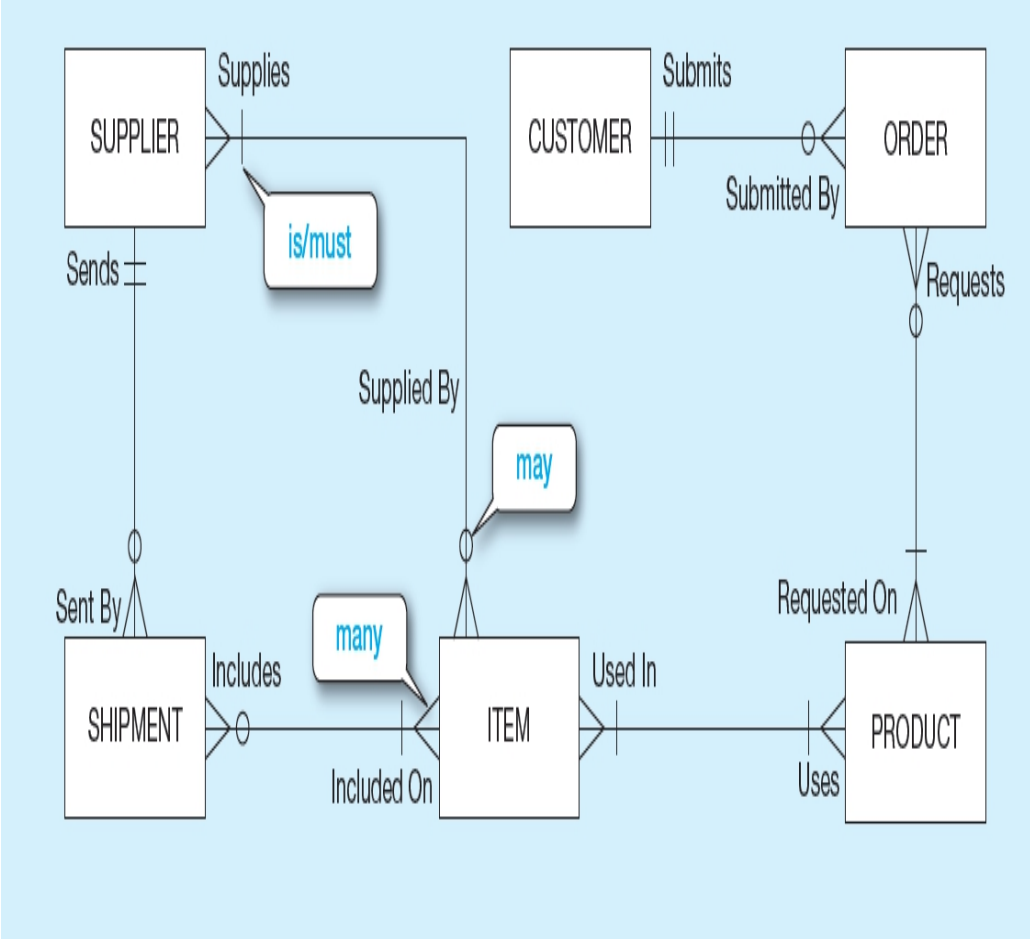
ANSI-SPARC Architecture and Database Design

- **Three phases of database design:**
  - **Conceptual database design**
  - **Logical database design**
  - **Physical database design.**



2. The Entity-Relationship Model

- An **entity** is a distinguishable object that exists.
  - a person, a place, an item, an event, or a concept in the user environment about which the organization wishes to maintain data
- Each entity contains a set of **attributes** - Properties or characteristics of an entity
- A **relationship** is an association among several entities.

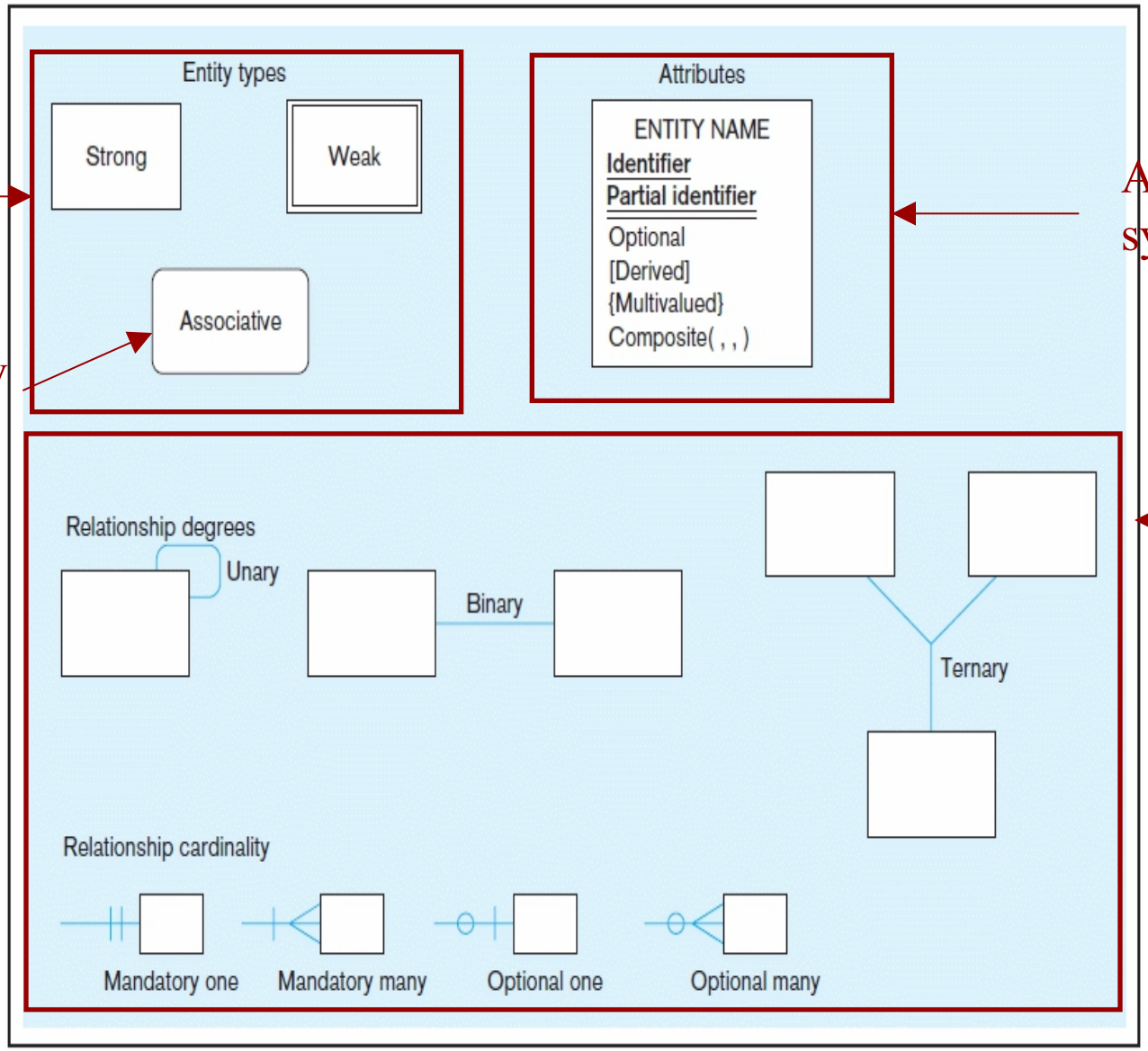


Basic E-R Model

Entity symbols

A special entity that is also a relationship

Relationship degrees specify number of entity types involved



Attribute symbols

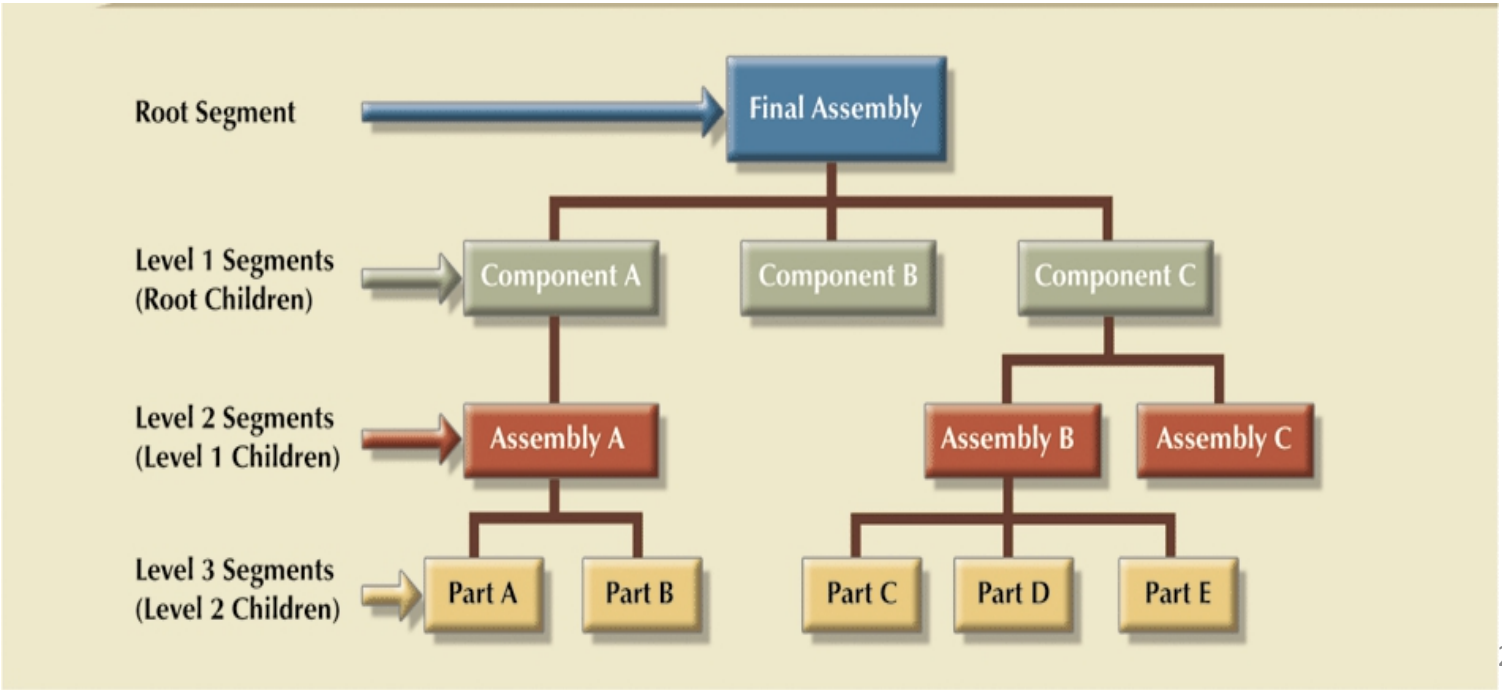
Relationship symbols

Relationship cardinalities specify how many of each entity type is allowed

3. Early Data Models

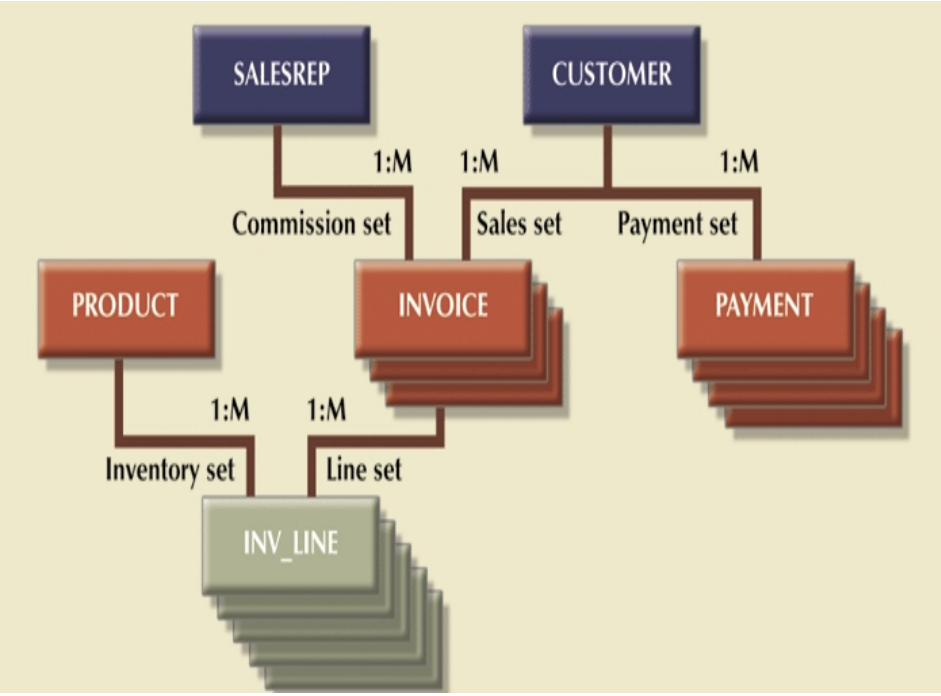
Hierarchical model

Represented by an upside-down "tree"
It contains Levels or Segments
It depicts a set of one-to-many (1:M) relationships between parent and its children segments.



Network model

Created to represent complex data relationships more effectively than the hierarchical model, to improve database performance and to impose database standard.



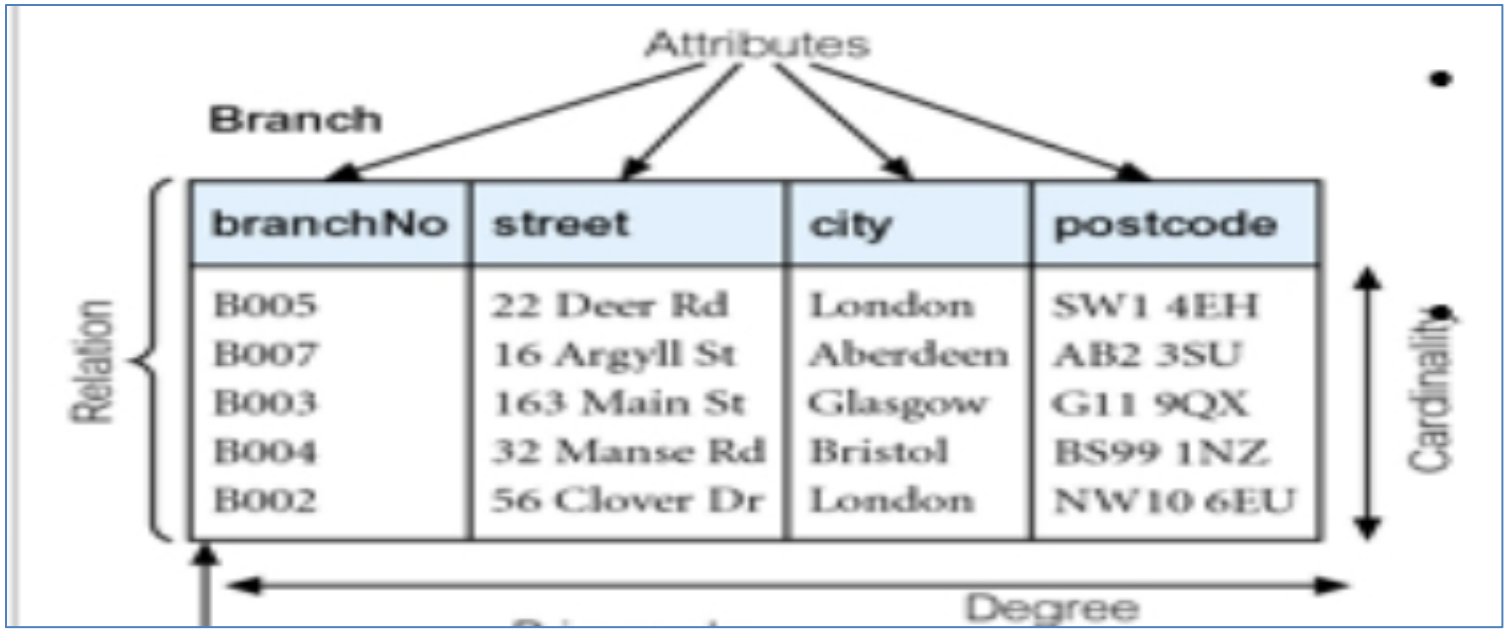
supports many-to-many relationship

Limitations of hierarchical and network data models

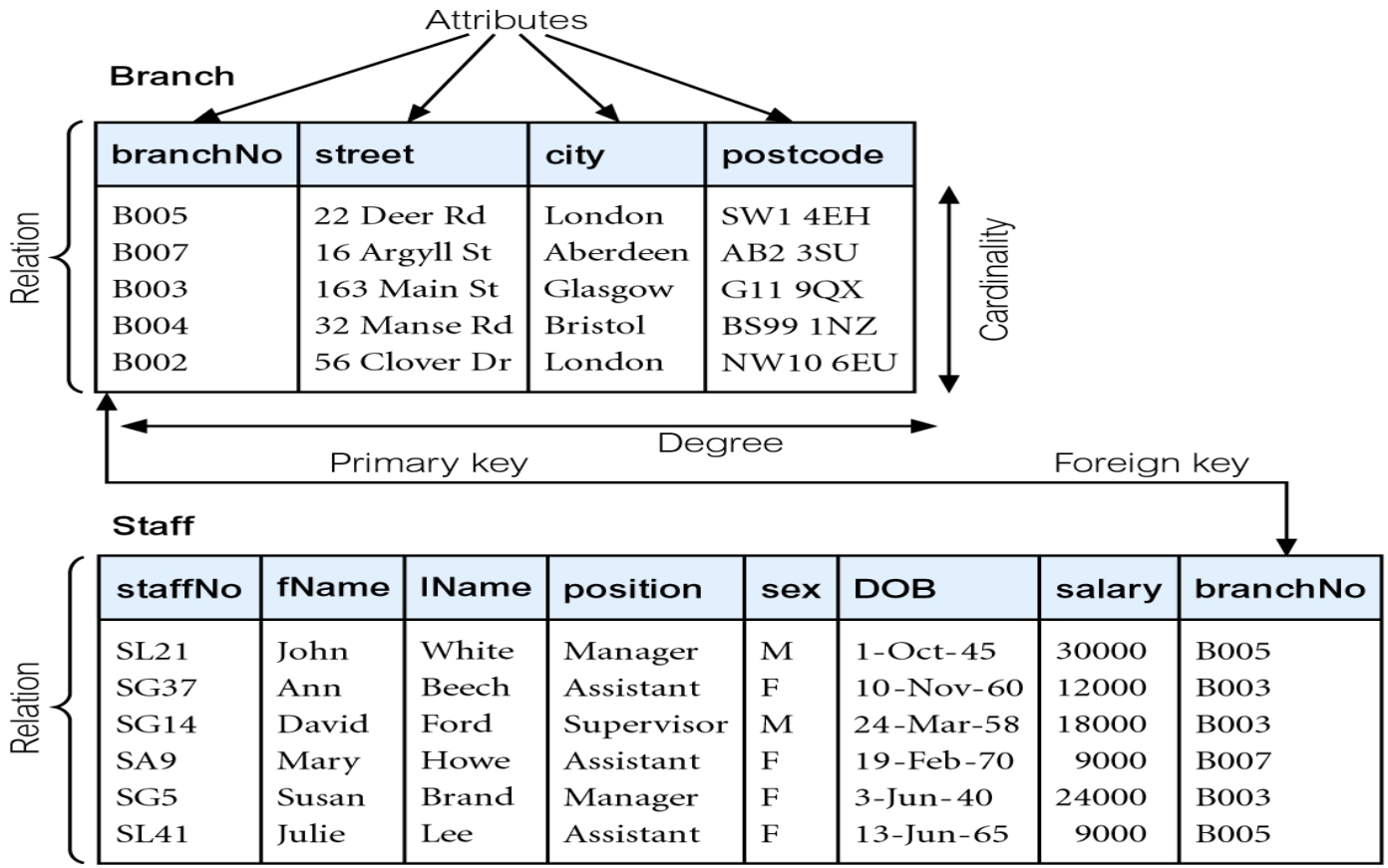
- **data are stored in rigid, predetermined relationships.**
- **no DDL existed, changing the structure of the data was difficult.**
- **lacked a simple query language, which hindered application development.**

4. Relational Model

- Data and relationships are represented by a collection of tables.
- Each table has a number of columns with unique names, called as attributes.
- Degree is the number of attributes in a relation.
- Cardinality is the number of tuples in a relation.



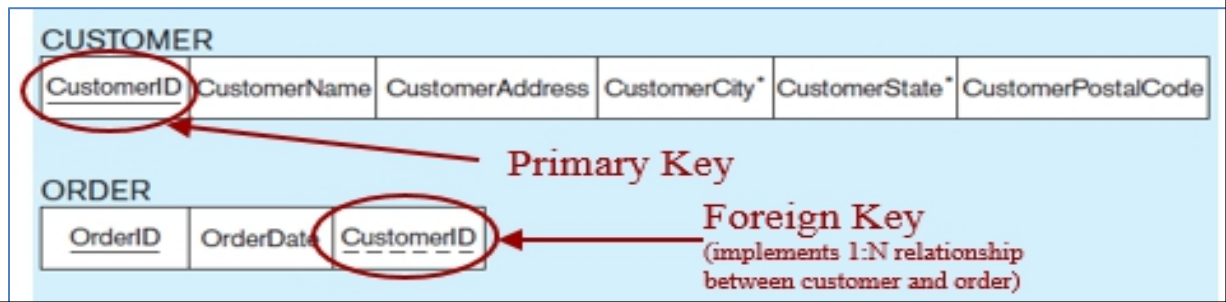
- **User sees the DB as a collection of tables in which data is stored.**
- **Each table is independent from another.**
- **Rows in different tables are related based on common values in common attributes**



Relational Model - Integrity Constraints

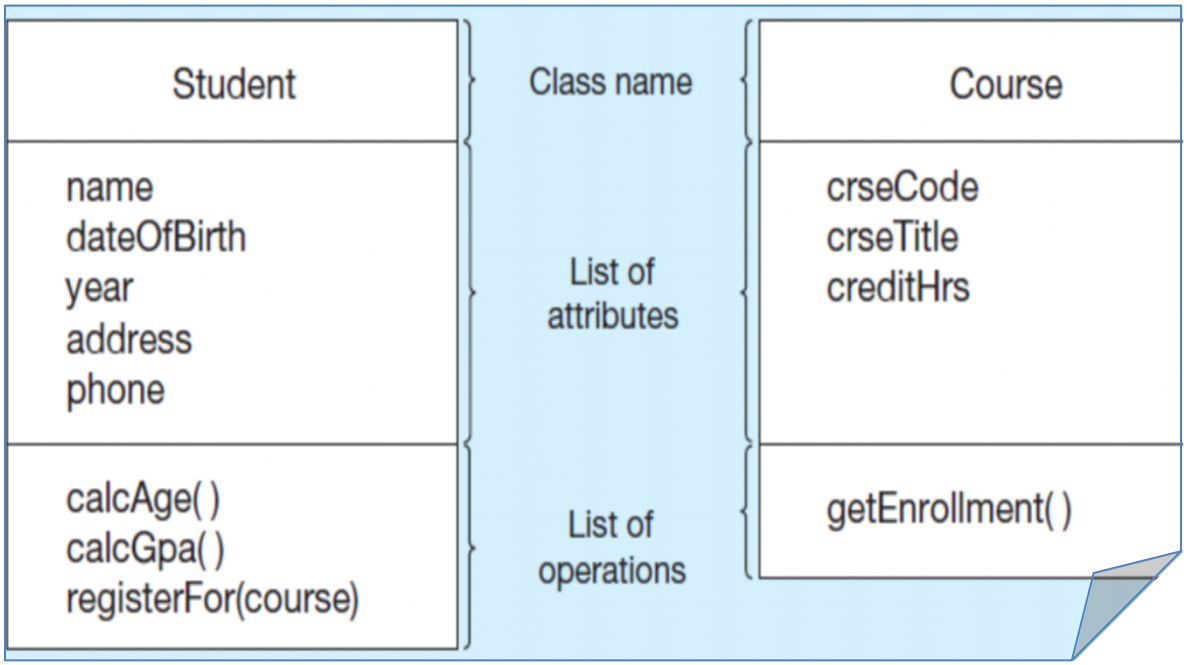
- Three basic types
  - Entity integrity, not allowing multiple rows to have the same identity within a table.
    - ✓ Primary key
  - Domain integrity, restricting data
    - ✓ predefined data types, e.g.: dates.
    - ✓ Set allowable values

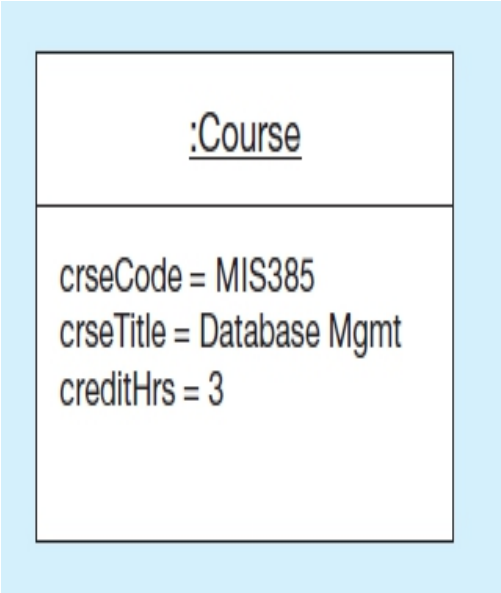
      Eg CREATE DOMAIN PriceChange AS DECIMAL
      CHECK (VALUE BETWEEN .001 and .15);
  - Referential integrity, requiring the existence of a related row in another table, e.g. a customer for a given customer ID.
    - ✓ Foreign key



5. Object-Oriented Model

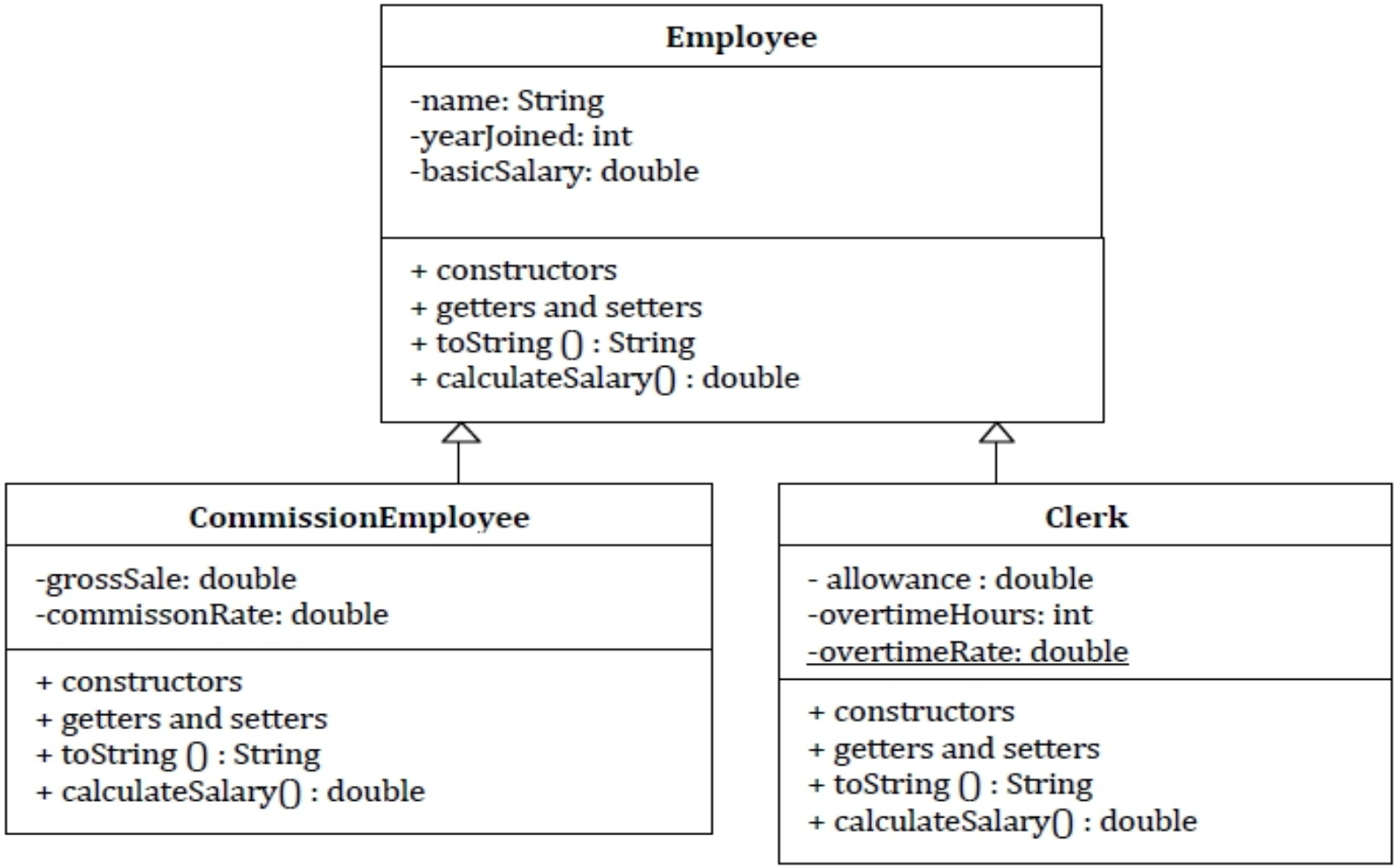
- Object is an abstraction of a real-world entity
- An object includes information - attributes/state, operations/behavior, and identity.
- Attributes/state describe the properties of an object
- Behavior/operation specity how an object acts and reacts
- Objects that share similar characteristics are grouped in classes





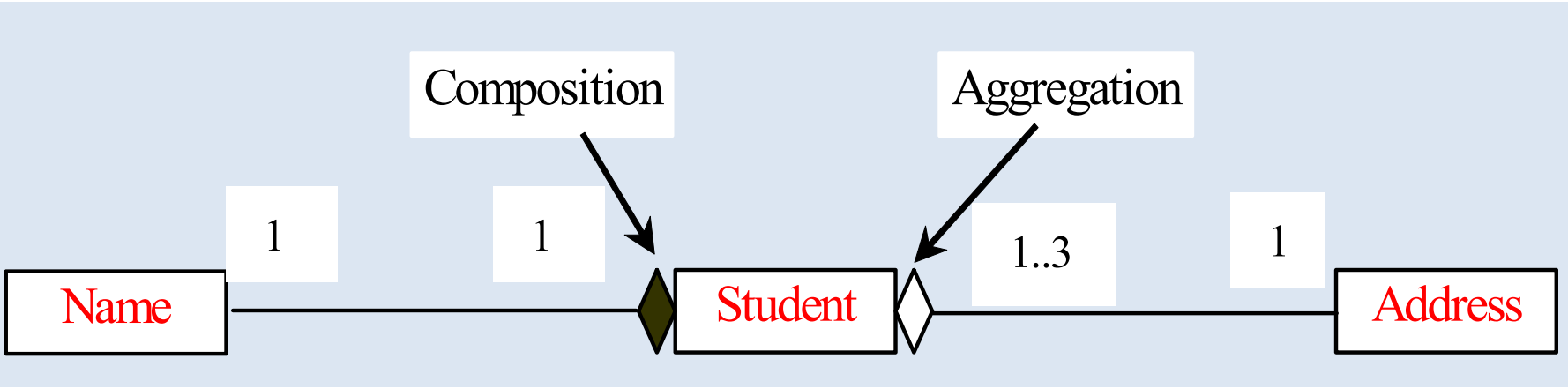
Object-Oriented Model

- Classes are organized in a ***class hierarchy.***
- **Inheritance**: object inherits methods and attributes of parent class



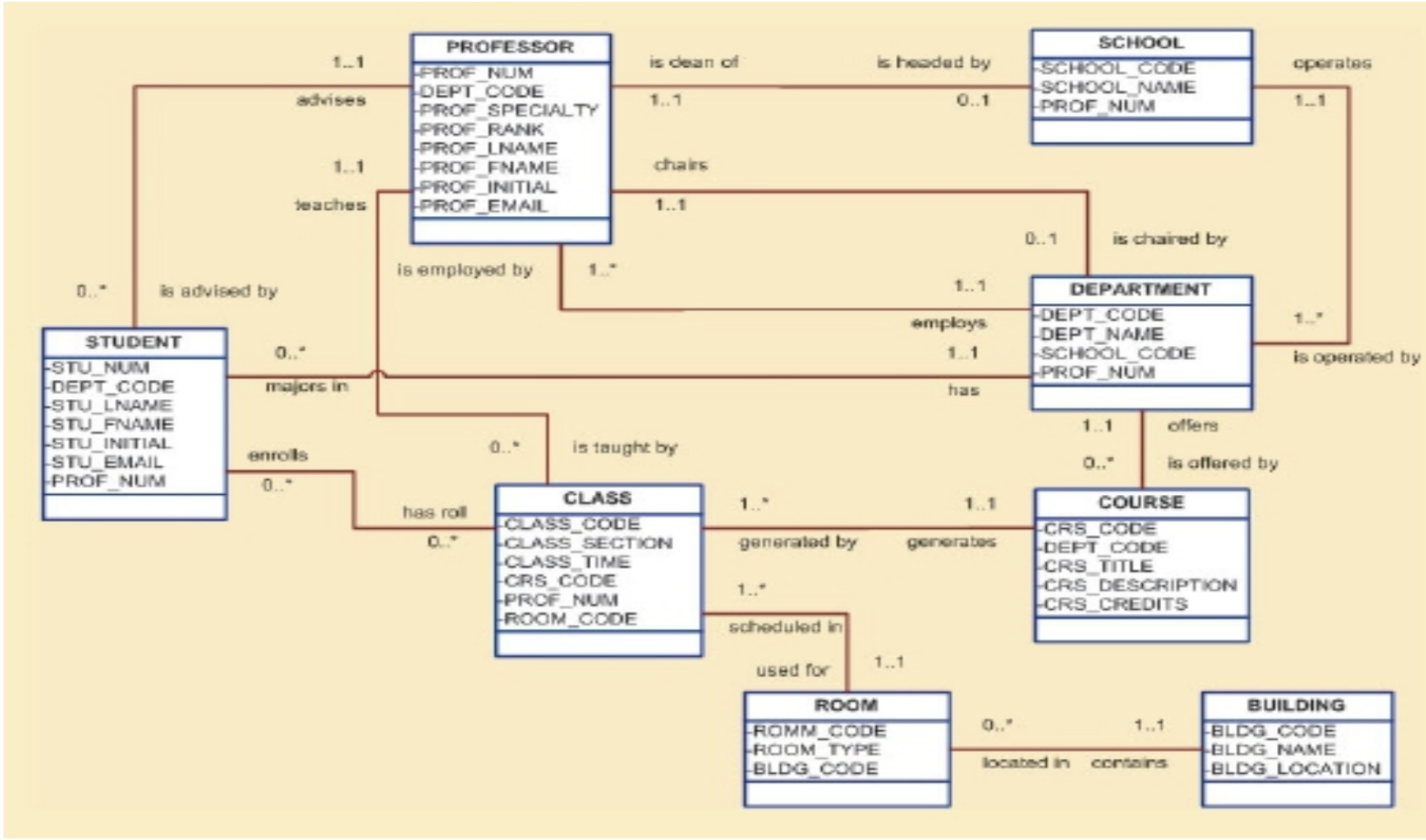
Object-Oriented Model

Aggregation and Composition



UML

Object-oriented modeling is commonly represented using UML



Why OO?

- Conventional data models (e.g., relational) are inadequate
  - Cannot model complex data and unstructured data
  - Cannot model processes (dynamic behaviour)
  - Does not support reuse

- OO model is a more 'natural' way to represent the real world

6. Semi-Structured Data Model

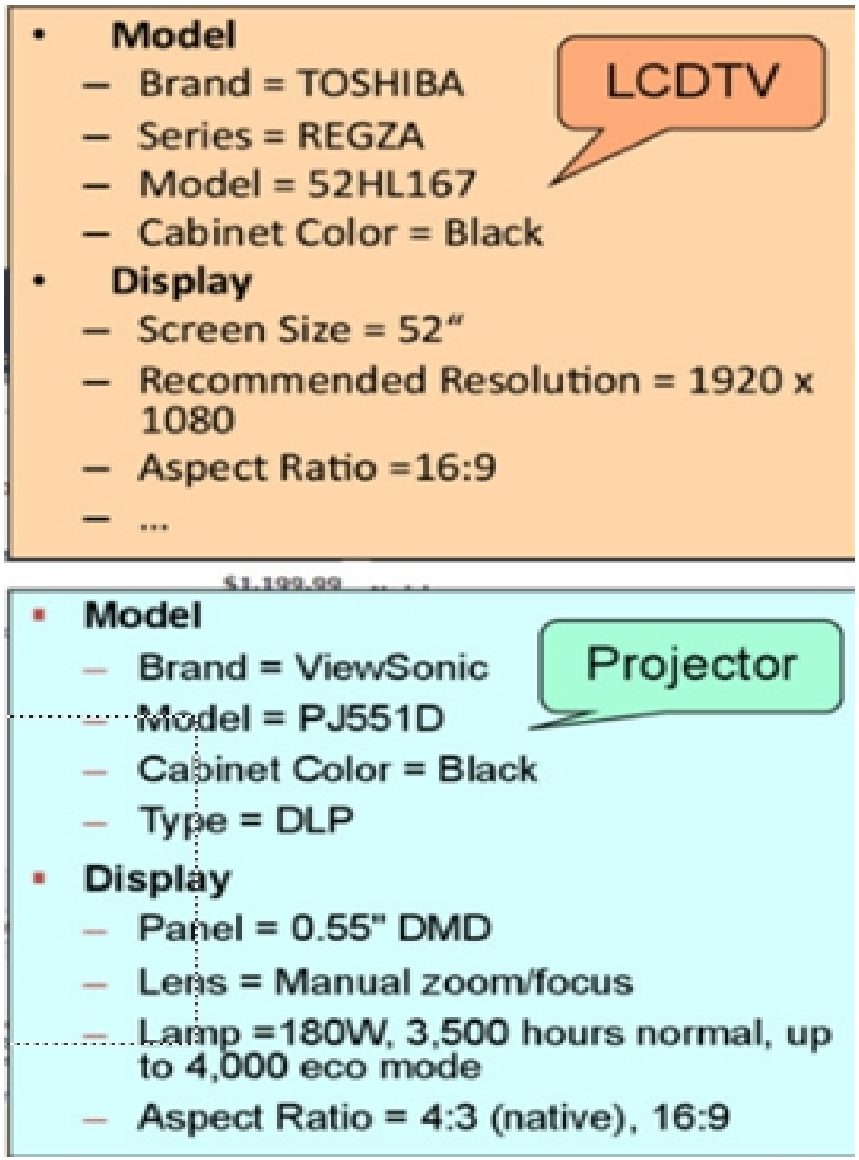
- **Structured data**
  - Represented in a **strict format**

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU

- **Semi-structured data**
  - Has a certain structure
  - **structure may not be rigid, regular, or complete**
  - **Eg. scientific data, E-Catalogs**
- **Unstructured data**
  - Very limited indication of the type of data
  - **Eg. text document** containing maps, images, sound, and video segments

Semi-Structured Data Model

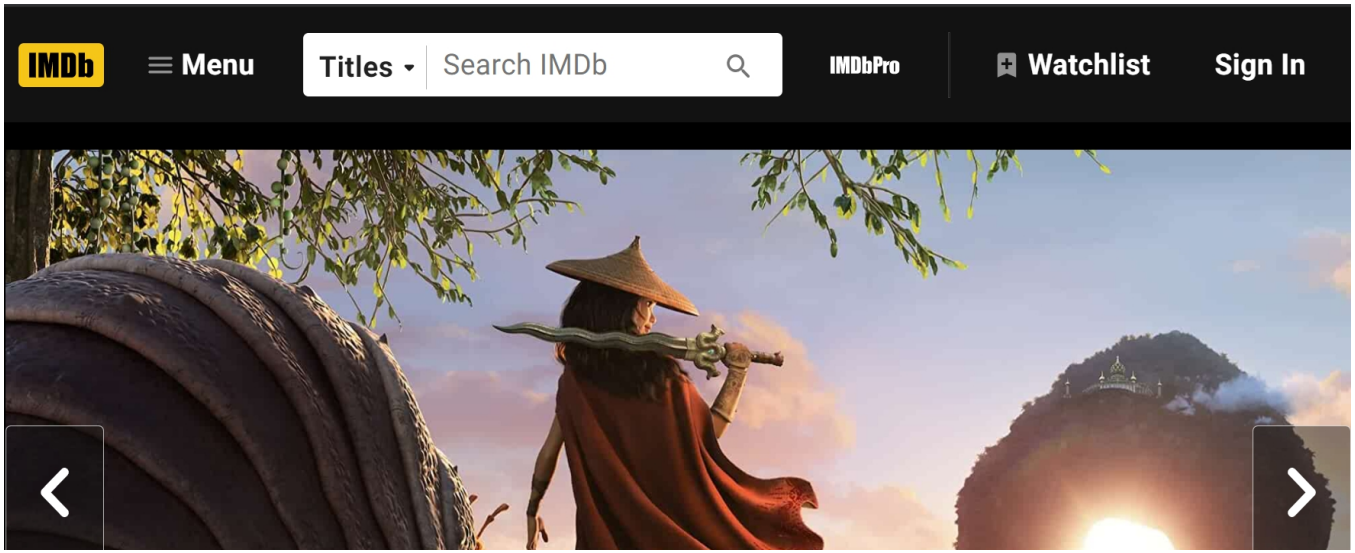
- **Semi-structured data**



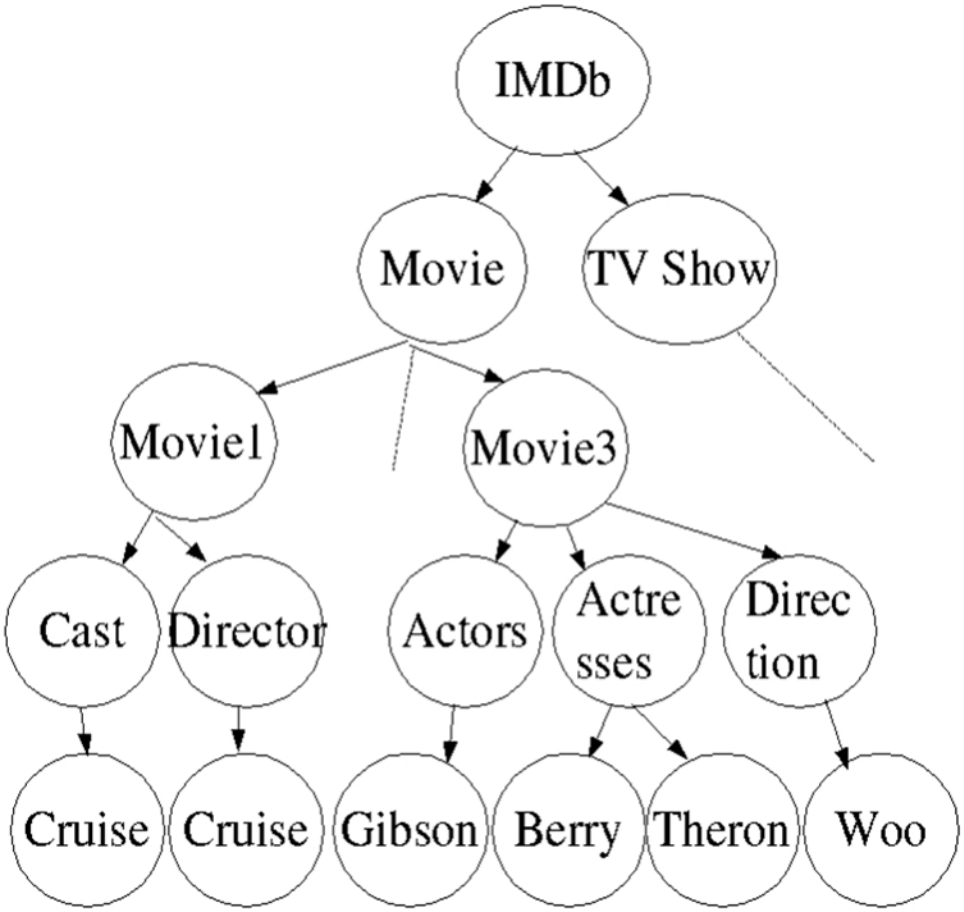
IMDb – A Motivating Example

- The Internet Movie Database is a classical example of a collection of semi-structured data
- Although the information pertaining to different movies may be essentially similar, their structure may be different!

www.imdb.com

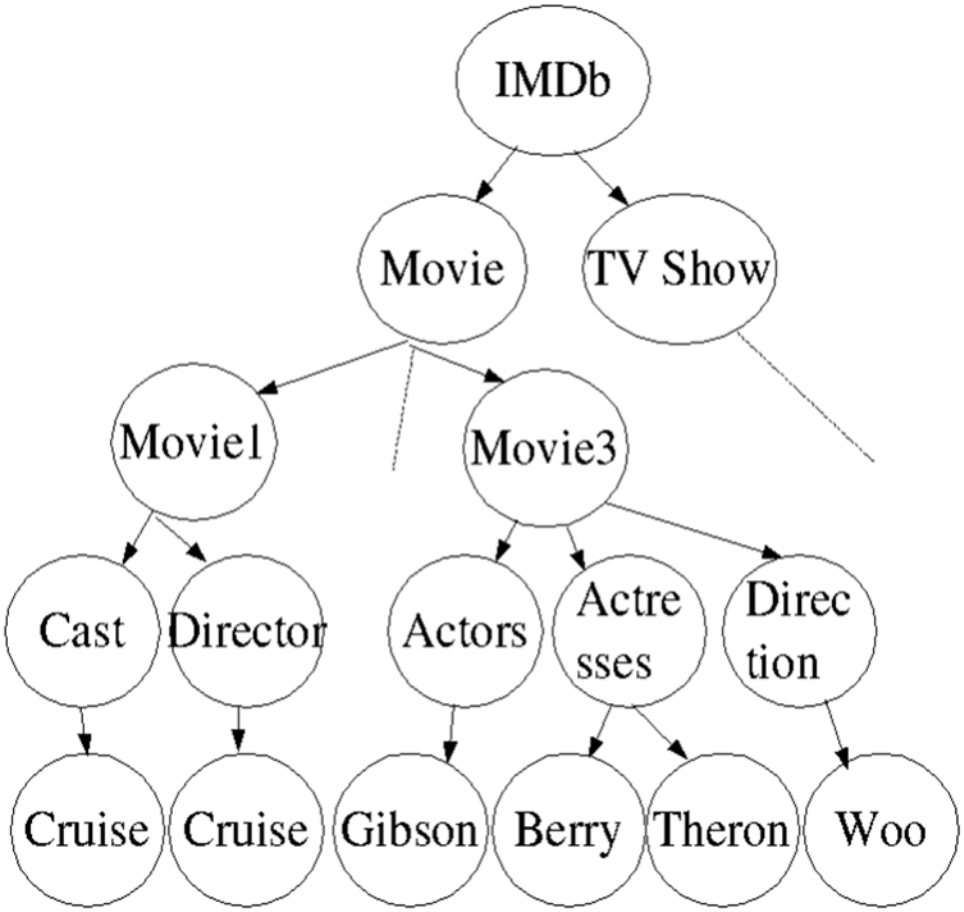


Irregularity In Structure



Example: Some movie may annotate information about the actors, choreographer, director and producer, while another movie may annotate additional information about the lyricist and the music director

Irregularity In Structure



Example: An actor's name may be represented as a

- **string or**
- **as a tuple (first_name, last_name)**

When data are added to the database dynamically, the structure of the database as a whole, also keeps changing

Semi-Structured Data Model

JSON, XML, other markup languages, email are all forms of semi-structured data.

XML Example

```
<products>
        <product>
                <number>1</number> <name>Zoom X</name> <price>10.00</price>
        </product>
        <product>
                <number>2</number> <name>Wheel Z</name> <price>7.50</price>
        </product>
        <product>
                <number>3</number> <name>Spring 10</name> <price>12.75</price>
        </product>
</products>
```

JSON & JSON Data Schema

```
{ "id": 1,
  "name": "Foo",
  "price": 123,
  "tags": {
      "Bar",
      "Eek"
  ],
  "stock": {
      "warehouse": 300,
      "retail": 20
  }
}
```

JSON Data Schema

```
{
  "$schema": "http://json-schema.org/schema#",
  "title": "Product",
  "type": "object",
  "required": ["id", "name", "price"],
  "properties": {
    "id": {
      "type": "number",
      "description": "Product identifier"
    },
    "name": {
      "type": "string",
      "description": "Name of the product"
    },
    "price": {
      "type": "number",
      "minimum": 0
    },
    "tags": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "stock": {
      "type": "object",
      "properties": {
        "warehouse": {
          "type": "number"
        },
        "retail": {
          "type": "number"
        }
      }
    }
  }
}
```

JSON data schema

- **defines the structure of JSON data.**
- **define validation, documentation, hyperlink navigation, and interaction control of JSON data**

Semi-Structured Data Model

- **The advantages of this model :**
  - **can represent the information of data sources that cannot be constrained by schema.**
  - **It provides a flexible format for data exchange between different types of databases.**
  - **The schema can easily be changed.**

http://en.wikipedia.org/wiki/Semi-structured_data
http://en.wikipedia.org/wiki/Semi-structured_model

Comparison with Relational Data

- **Inefficient: tags which represent schema information, are repeated**
- **Better than relational tuples as a data-exchange format**
  - **–Unlike relational tuples, XML/JSON data is self-documenting due to use of tags**
  - **–Non-rigid format: tags can be added ie schema can easily be changed.**
  - **- Allows nested structures**
  - **–Wide acceptance, not only in database systems, but also in browsers, tools, and applications**

```
{ "id": 1,
  "name": "Foo",
  "price": 123,
  "tags": {
        "Bar",
        "Eek"
   ],
   "stock": {
        "warehouse": 300,
         "retail": 20
   }
 }
```

References

- ***Database Systems: A Practical Approach to Design, Implementation and Management*.** **Connolly, T. M. and Begg, C. E. .**
- ***Modern Database Management.*** **Hoffer, J.A., Prescott, M., and McFadden,F.**
- ***Database System Concepts*** **. Silberschatz, A., Korth, H,. and Sudarshan, S.**
- ***Fundamentals of Database Systems.*** **Elmasri, R. and Navathe, S.B.**