DATABASES

Unit 2-1. Data Models

José A. Cruz-Lemus, Marcela Genero

Goals

- Showing the **elements** which are part of a **data model** (DM).
- Distinguishing amongst the main **abstraction mechanisms**.
- Analyzing the types of **restrictions** in a DM.
- **Classifying** the most used DM.

Agenda

- Basic concepts
- Data model definition
- Main abstraction mechanisms
- Integrity restrictions
- Data models taxonomy
- DM in the process of designing a DB and an IS
- Physical design
- Data models examples

Bibliography

- Tecnología y Diseño de Bases de Datos
  - Piattini, Marcos, Calero, Vela (2006) Ra-Ma. Tecnología y diseño de bases de datos. RA-MA.
    - Chap. 4

- "A model is a mental construction based on reality in which the main components and relationships of the piece of reality analyzed are reproduced".

- 2 meanings:
  - **Simplified reproduction** of reality (empirical sciences)
  - **Reality** itself (painter)

- "***Abstraction*** mechanism which allow us see the forest (i.e., the information contained in data) as opposed to the trees (i.e., individual data values)", Tsichritzis y Lochovsky (1982).
- *"Set of* ***conceptual tools for describing*** *the representation of the information in terms of data. Data models comprise aspects related to: data structures and types, operations, and restrictions"*, Dittrich (1994).
- "Set of ***concepts, rules, and conventions*** which allow describing and handling data in the part a of a certain real world which we want to store in a database", De Miguel *et al.* (1999).

DL= DM+ Syntax

Examples:

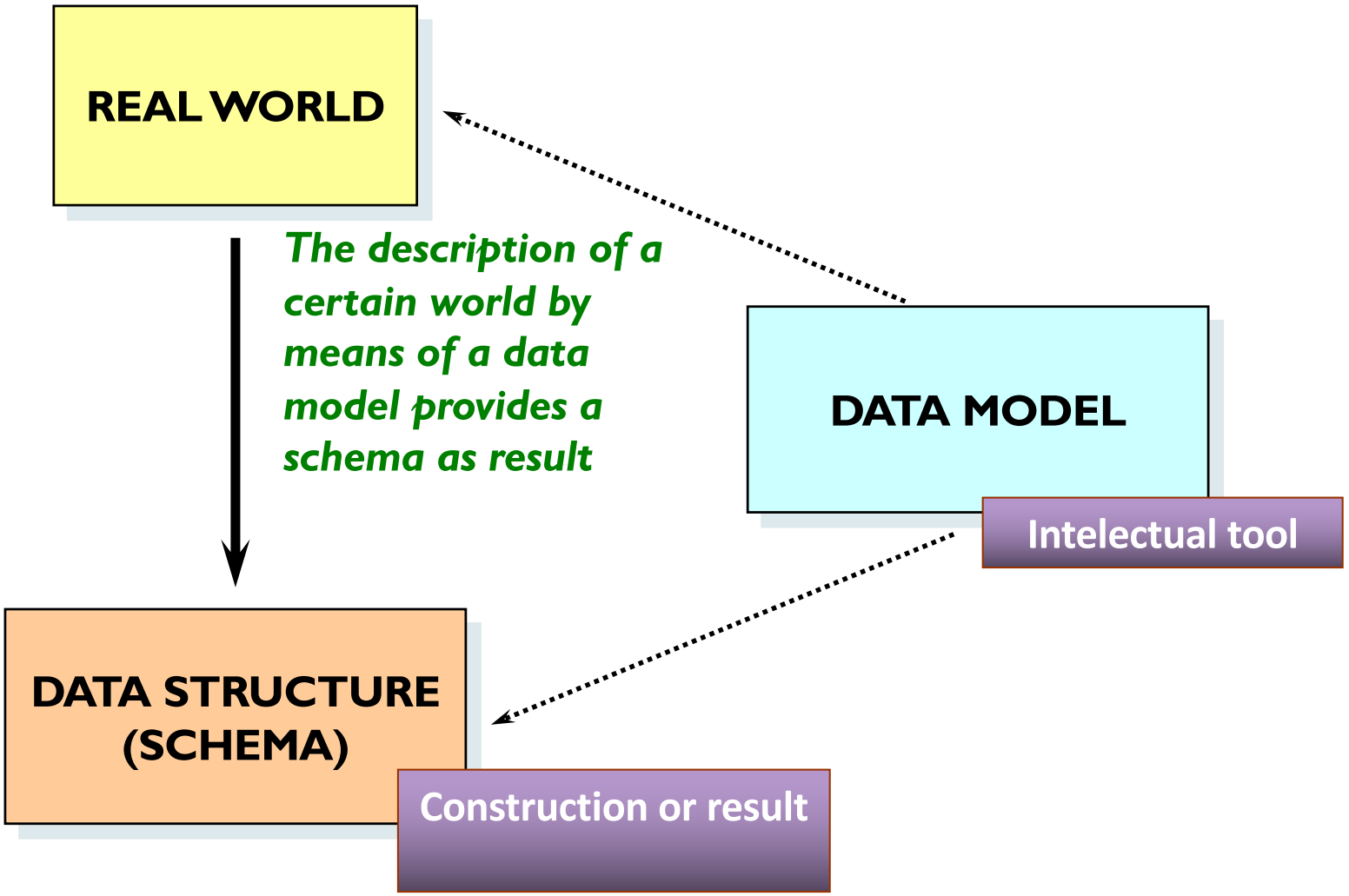
SQL = Relational Model + Syntax

QBE = Relational Model + Syntax (different)

OQL = Object Model + Syntax

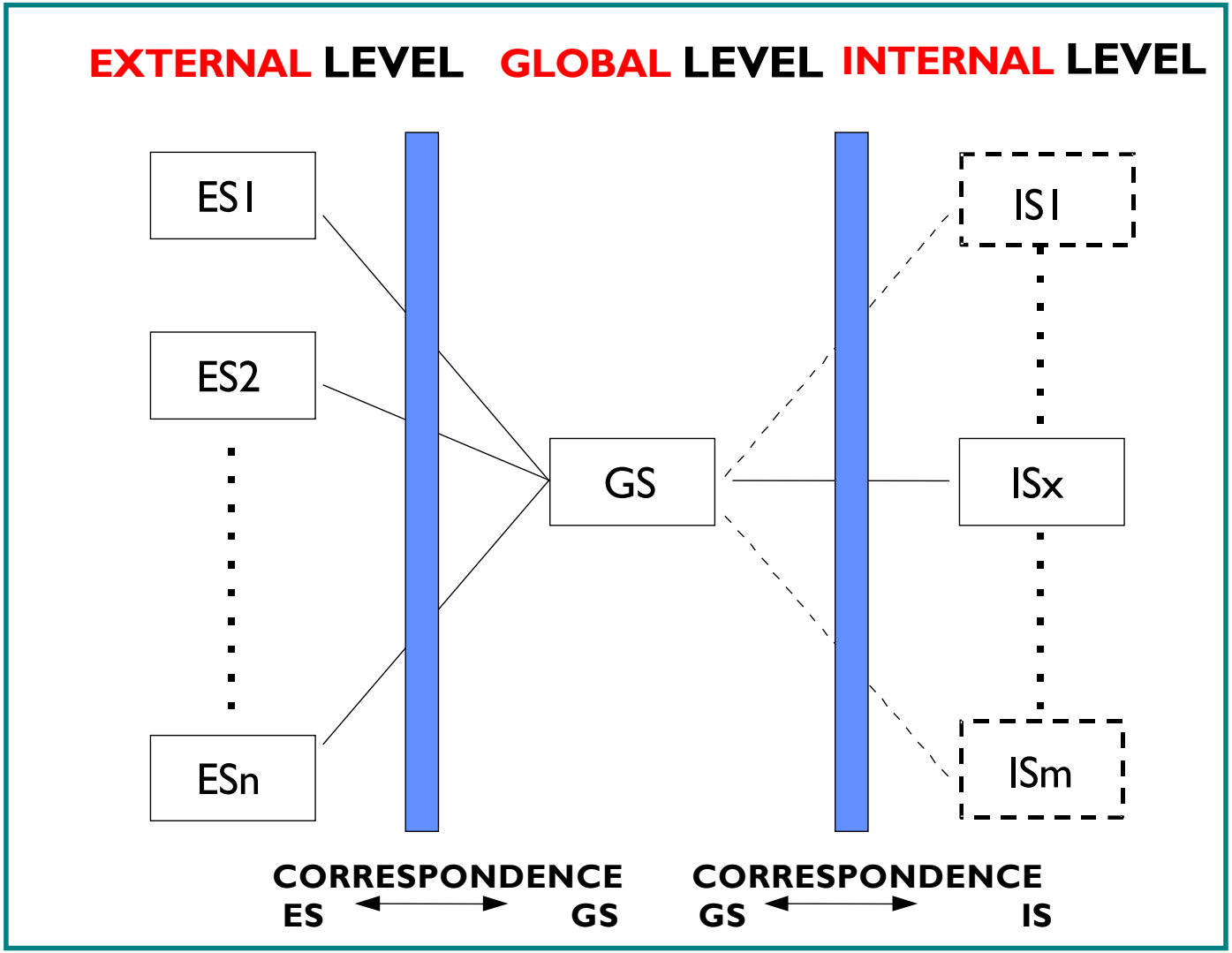
- "A (data) schema is the specific ***description*** of a certain ***mini-world*** *in terms of a data model*. A database is the data collection representing information about this mini-world", Dittrich (1994).

- "**Representation** of a certain real world (universe of discourse – UoD) in terms of a data model", de Miguel, Piattini y Marcos (1999).


REAL WORLD
The description of a certain world by means of a data model provides a schema as result
DATA MODEL
Intelectual tool
DATA STRUCTURE (SCHEMA)
Construction or result

- A **specimen** (***ejemplar*** in Spanish) **of an element** of a schema is referred to the **data** stored in that element from that schema in a certain moment.

- We will use the expression "database" in the abstract meaning of all possible models. If we need to refer to its content in a certain moment, we will use the term *specimen* or "the database in the *i* moment (BD_i)".

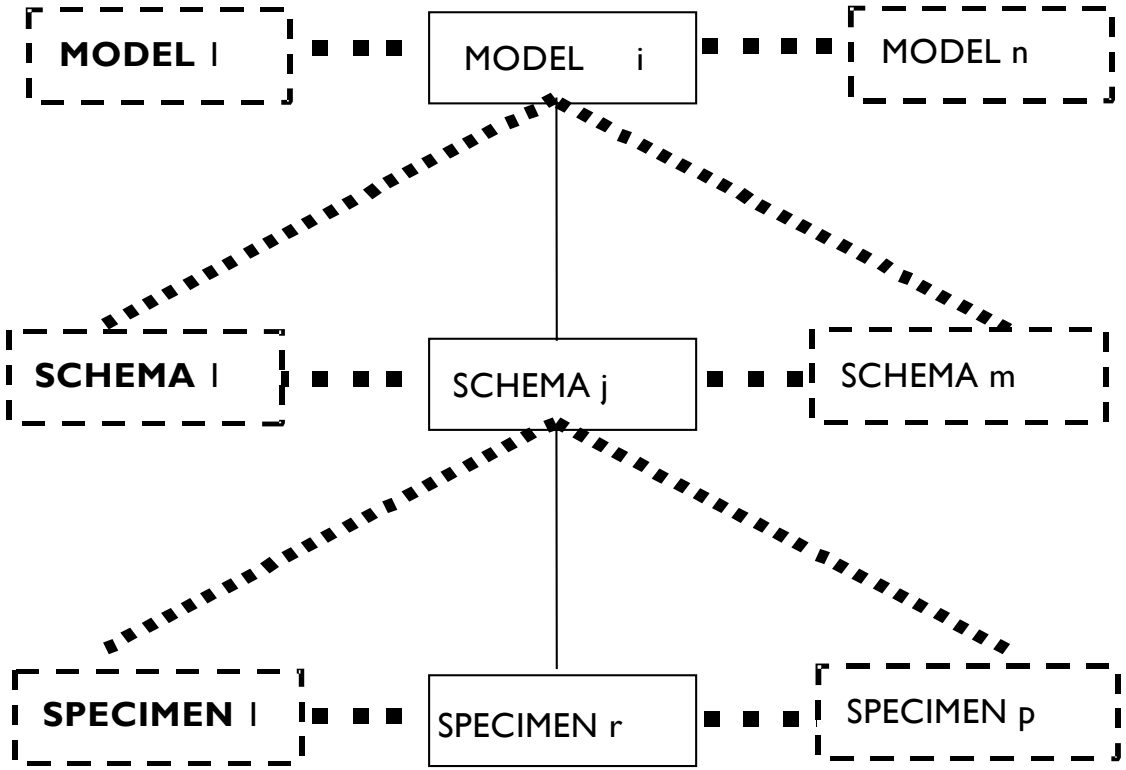


Schemas for the three levels in ANSI architecture

Set of rules for structuring real world data

Perception of a certain reality interpreted according to a certain model

Values which take the perception of a certain reality (schema) in a time point



Data model definition

- Although there are lots of DM, it is possible to abstract some common characteristics. Thus, the concept of data model can be defined as:
  - "***a set of well-defined concepts, rules and conventions which allow us to apply a set of abstractions for describing and manipulating the data of a certain real world which we want to store in a database***".

- DM make easier the creation of categories through applying abstraction mechanisms (classification, aggregation, etc.). This way, we can find two different types of models (the same happens in programming languages) :
  - Strongly typed, ← (used for DB).
  - Weakly typed.

Data model definition
Properties

- STATIC
  - Permitted elements
    - Objects
    - Associations
    - Object characteristics
    - Domains
  - Not-permitted elements (restrictions)
    - Inherent
    - Integrity or semantics
- DYNAMIC
  - Set of operators
  - Each operator has two components:
    - Locator
    - Action

- A DM must provide capabilities for collecting the static and dynamic aspect of the reality. It is defined as:

$$DM = \langle G, O \rangle$$

  - where **G** is the set of generation rules for representing the **static component**, i.e., describing the **structures** of the UoD, and
  - **O** is the set of authorized **operations** on the corresponding structure. These operations will be used for representing the **dynamic component**.

- The static component of a given MD expressed with a specific syntax is the ***Data Definition Language*** (DDL), and the dynamic component the ***Data Manipulation Language*** (DML); both together constitute the ***Data Language*** (DL).
- DBMS usually have a *Query Language* (QL) and a *Control Language*.

- The permitted elements are not the same for every DM (terminology changes), but normally we can find:
  - **Objects** (entities, relations, records, etc.)
  - Object **associations** (relationships, sets, etc.)
  - Object or association **properties** (attributes, fields, data elements, etc.)
  - **Domains**, a set of nominated homogenous values over which properties are defined.

- The abstractions acknowledged by the model shall be applied on these permitted elements.
- The **representation** of these elements depends of each DM. We can find **graphs** (E/R, UML) o **tables** (Relational Model).

- Not permitted elements are known as **restrictions**.
- Types:
  - **Inherent** restrictions (model's):
    - They come from the very nature of the data model, which do not permit certain structures.
  - **Integrity or semantic** restrictions (user's):
    - They permit catching the semantics of the UoD to be modelled and verifying the correctness of the data stored in the DB.
    - According to the instruments given by the data model for defining and managing restrictions, they can be:
      - **DM's own**: defined by the designer but managed by the data model, which recognizes and puts them into the schema.
      - **External to the DM**: full reasonability of the designer, as the data model cannot recognize them nor provides instruments for using them.

- The **dynamic component** is formed by a set of **operators** defined on the structure of the corresponding DM, as not every structure permit the same kind of operation.

- Applying an operator on a schema specimen will transform it into another specimen:

  O [DBi] = DBj

- Both DBi and DBj must be valid DB specimens, i.e., the values in both of them must belong to any of the categories defined in the schema and also they must fulfil the integrity restrictions (and any possible restriction regarding the change of state, if necessary).

We use abstraction mechanisms continuously:

- AMB-0978 vehicle is an ambulance.
- It is made of 4 wheels, a chassis, an engine, etc.
- An ambulance is a vehicle for collecting and transporting sick people.
- It is owned by the CUASER Company and driven by Manuel Fernández.

We use abstraction mechanisms continuously :

- **Classification**:
  - AMB-0978 vehicle is an ambulance.
- **Aggregation**:
  - It is made of 4 wheels, a chassis, an engine, etc.
- **Generalization**:
  - An ambulance is a vehicle for collecting and transporting sick people.
- **Association**:
  - It is owned by the CUASER Company and driven by Manuel Fernández.

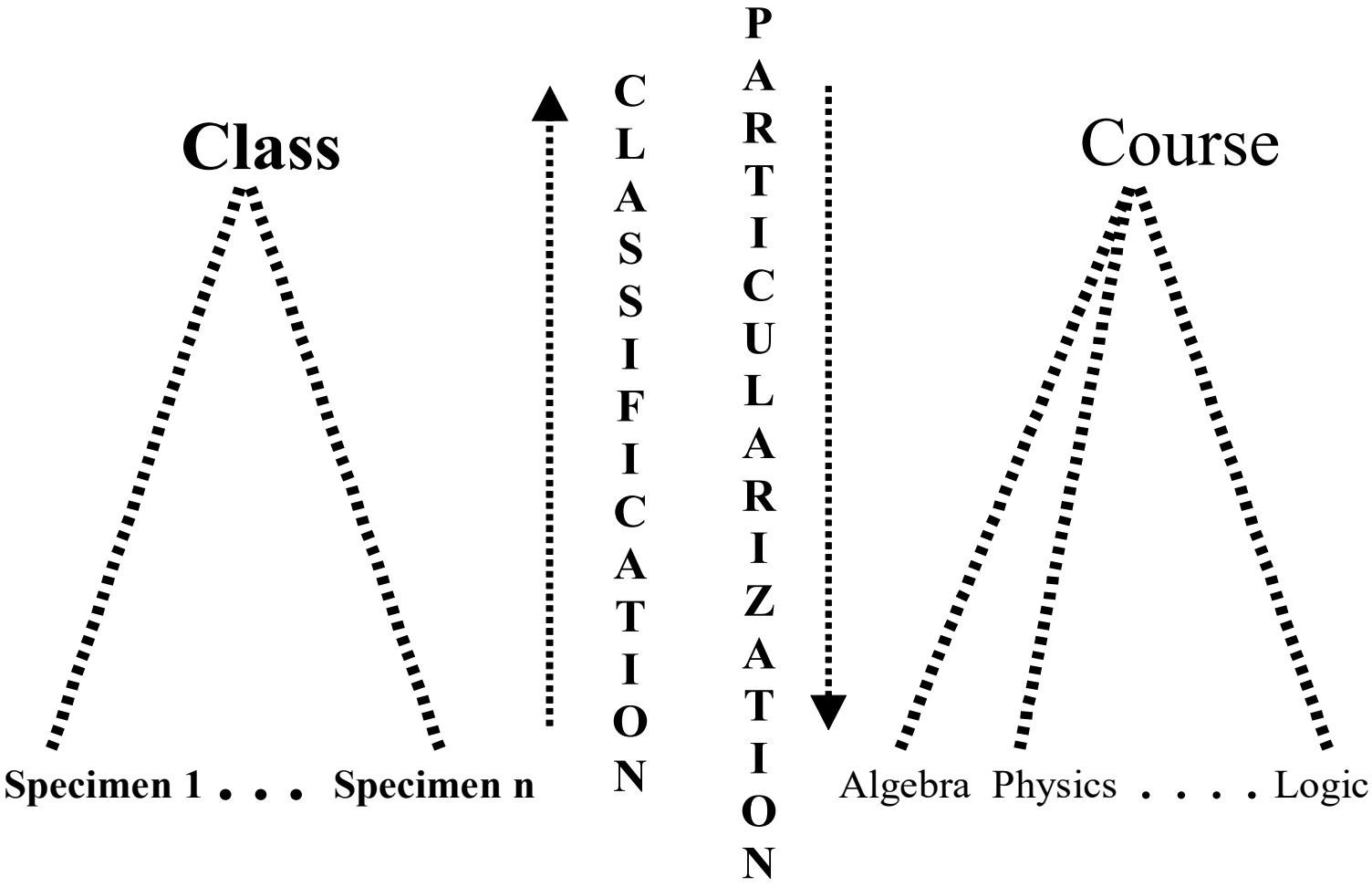
Abstraction Mechanisms
Classification

- **Classification** is the action of abstracting the characteristics common to a set of specimens for creating a category to which they belong.
- The opposite mechanism is **Particularization**.
- BRODIE (1984) defines classification as:
  - *An abstraction mechanism in which a set of objects are considered as a higher level class of objects.*
  - *A class of objects is a precise characterization of all the properties shared by all the objects in a collection.*
  - *An object is a specimen of an objects class if it has go the properties defined in the class.*
- Example: we call **vehicles** to any machine, animal or other self-propelled thing used for taking beings or objects from one place to another**.**
  - Ambulance => YES
  - Crane => NO (not self-propelled).

Abstraction mechanisms

Classification

- Classification is related to an element **belonging** to a set:
  - An **IS_MEMBER_OF** relationship can be found.
- Class specimens have similar characteristics and a class can be defied through these characteristics. They get specific values for each class specimen.
- Same objects admit different classifications. E.g., subjects can be classified as:
  - mandatory / optative
  - whole course long / one term long
  - first, second, third… course
  - Theoretical / applied
- Every DM related to databases permit classification.


Class
Course
CLASSIFICATION
PARTICULARIZATION
Specimen 1 . . . Specimen n
Algebra Physics . . . . Logic

- **Aggregating** is building a new model element as the **composition** of other elements (**components**):
  - An **IS_A_PART_OF** relationship can be found.

- The opposite mechanism is called **disaggregation**.

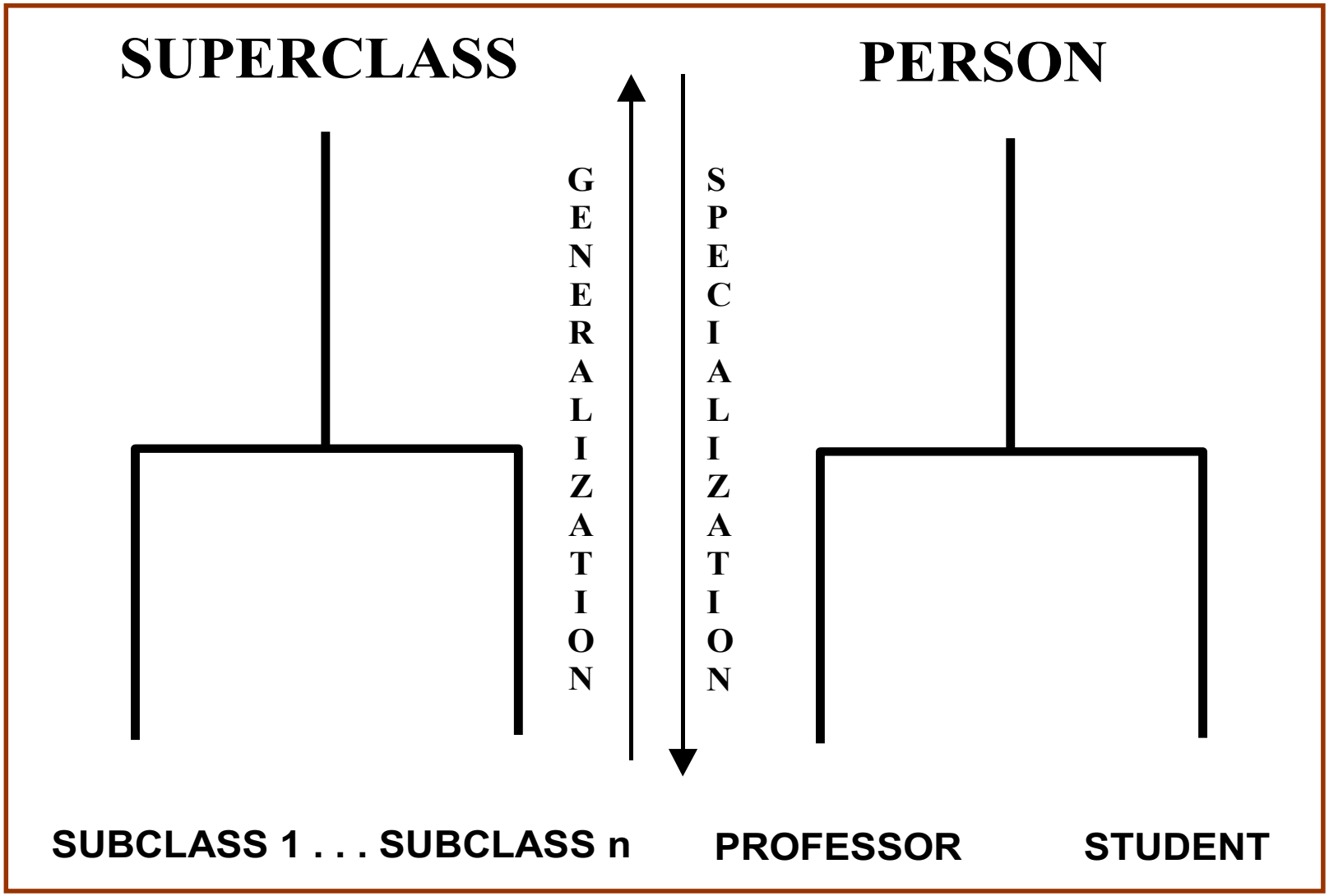
- There are three types of aggregation:
  - **Classes** aggregation for obtaining a **composed class**

    (included in semantic DM: E/R, UML, etc.)
  - **Properties** aggregation for obtaining a **class**

    (explicit or implicitly permitted by all DM)
  - **Properties** aggregation for obtaining a **composed property**

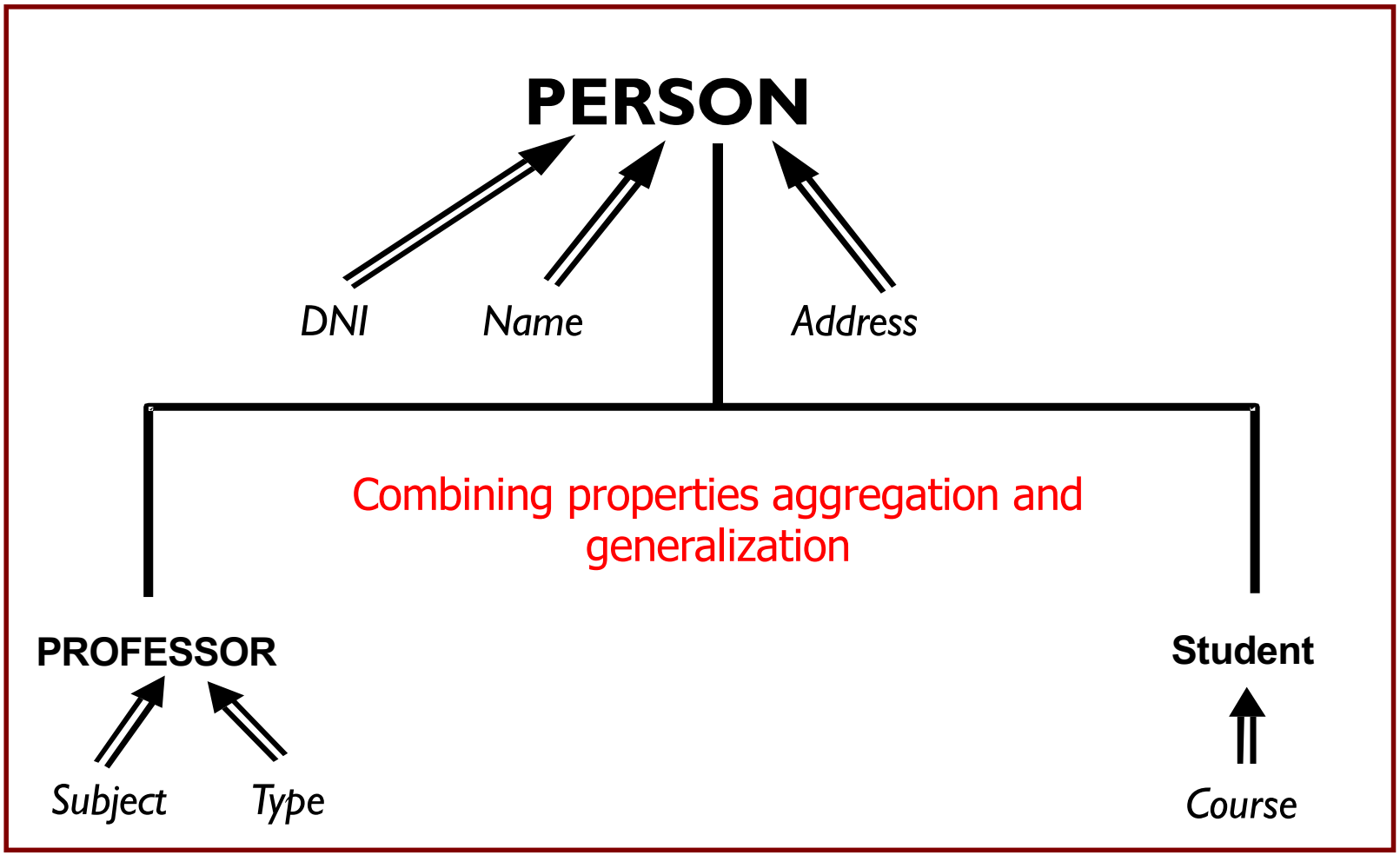
    (permitted by some DM: Codasyl YES, Relational NO)

- Generalization is the act of abstracting the common characteristics to several classes (**sub-classes**) for constructing a most general class (**super-class**) which comprises them:
  - The set of specimens of a sub-class 'is a' sub-set of the specimens of the corresponding super-class.
  - An **IS_A** relationship can be found.
  - E.g.: Super-class PERSON is a generalization of the sub-classes PROFESSOR and STUDENT.
- The inverse mechanism is **Specialization**.
- Every specimen of a sub-class is also a specimen of the super-class, so it has got all the specific characteristics of the sub-class and **inherits** all the characteristics of the super-class.
- Although this is an intuitive and useful mechanism, it is not used in many data models (e.g. Relational).


SUPERCLASS
PERSON
GENERALIZATION
SPECIALIZATION
SUBCLASS 1 . . . SUBCLASS n
PROFESSOR
STUDENT

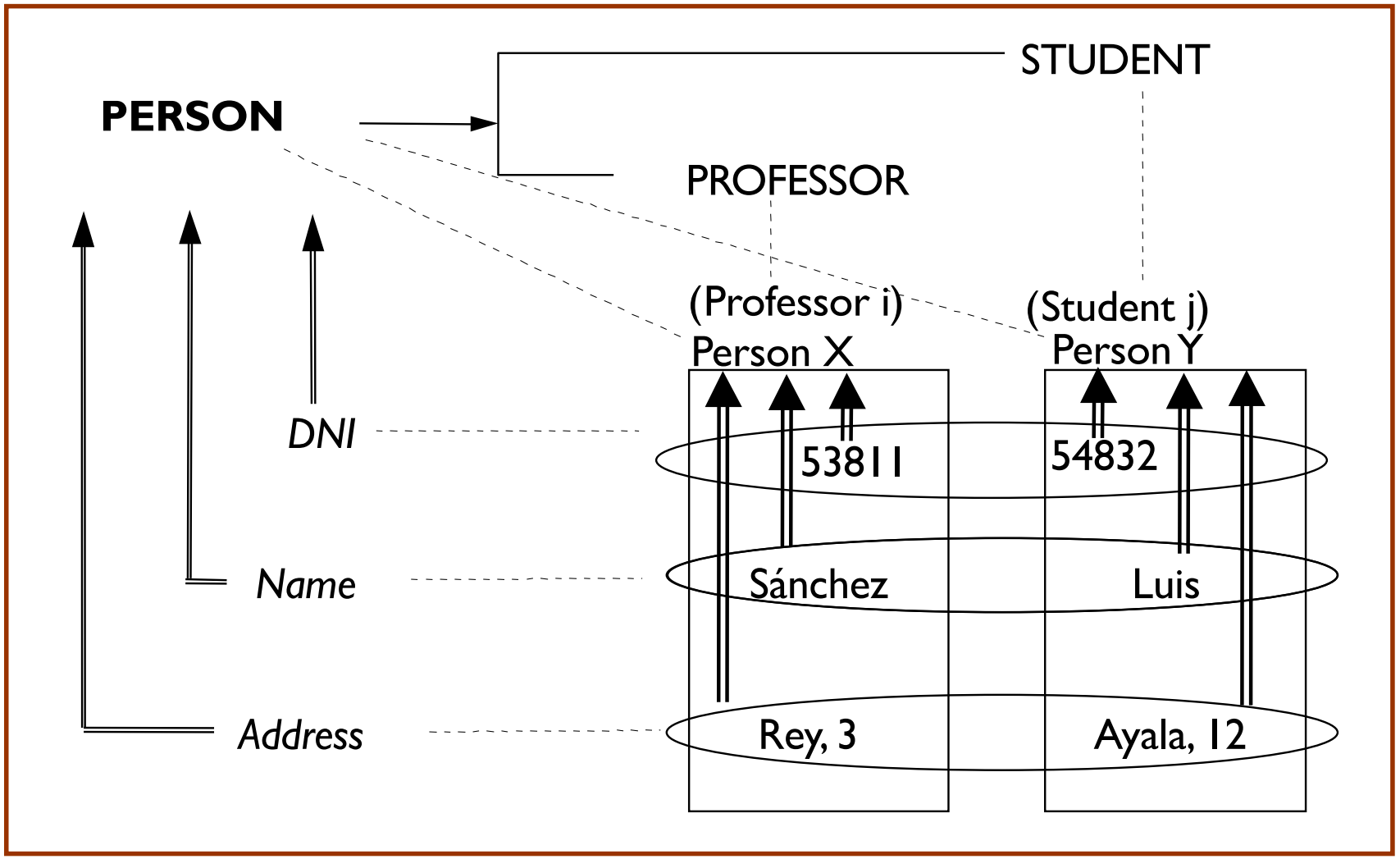
- **Association** is an abstraction mechanism used for relating 2 or more classes (and hence, their specimens), creating an element of a different kind
- It does not appear is some DM (e.g., Relational)
- Inverse mechanism is **Disassociation**.
- Some authors consider it as a special type of aggregation, but there are several differences (De Miguel et al., 1999):
  - When two or more categories are associated, the new element has certain characteristics that for distinguishing it from the normal categories, so that, in general, DM create a new concept for representing it.
  - This new element is *not composed* (as in aggregation) by the associated elements.
  - Aggregation may use inheritance, but association does not.

- Example

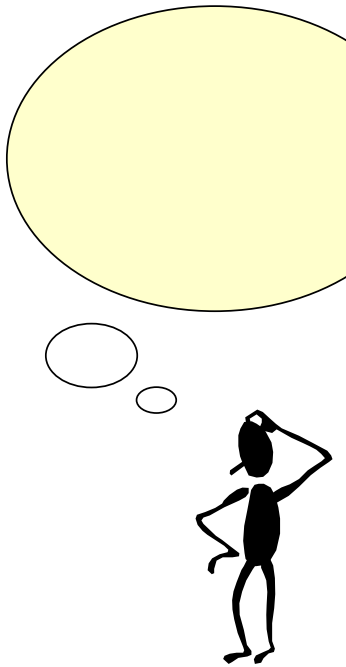


When modeling a certain reality, we will often need to combine several abstractions and form an abstraction hierarchy

Classification, aggregation and generalization



Integrity restrictions



.....

If BeginDate ≥ EndDate
THEN ERROR

.....

If ProfCode in COURSES= 'NULL'
THEN ERROR

.....

CHECK
BeginDate < EndDate

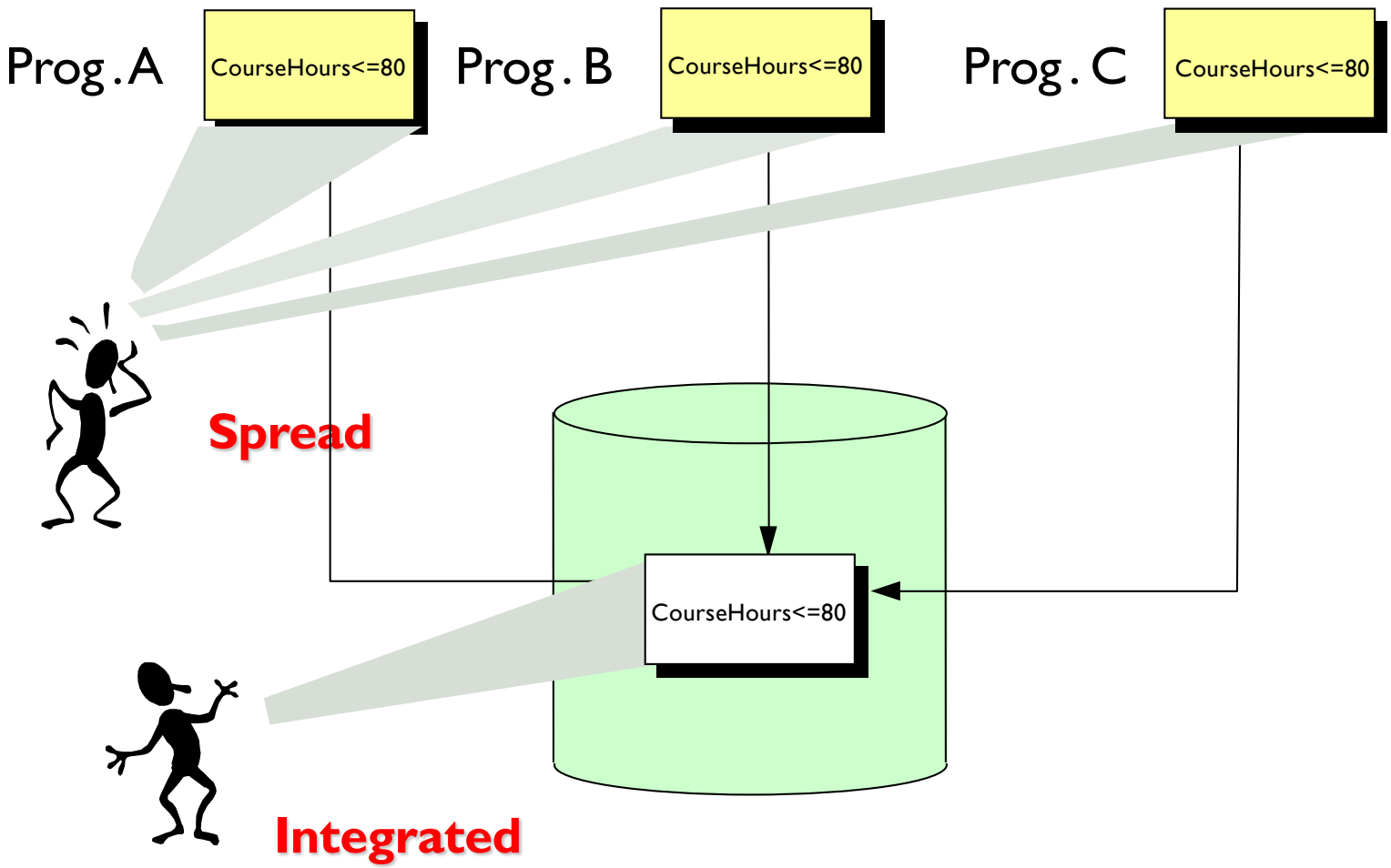
COURSES.ProfCode
Not Null

.....

USER PROGRAMS DATABASE

Data semantics migration

Integrity restrictions

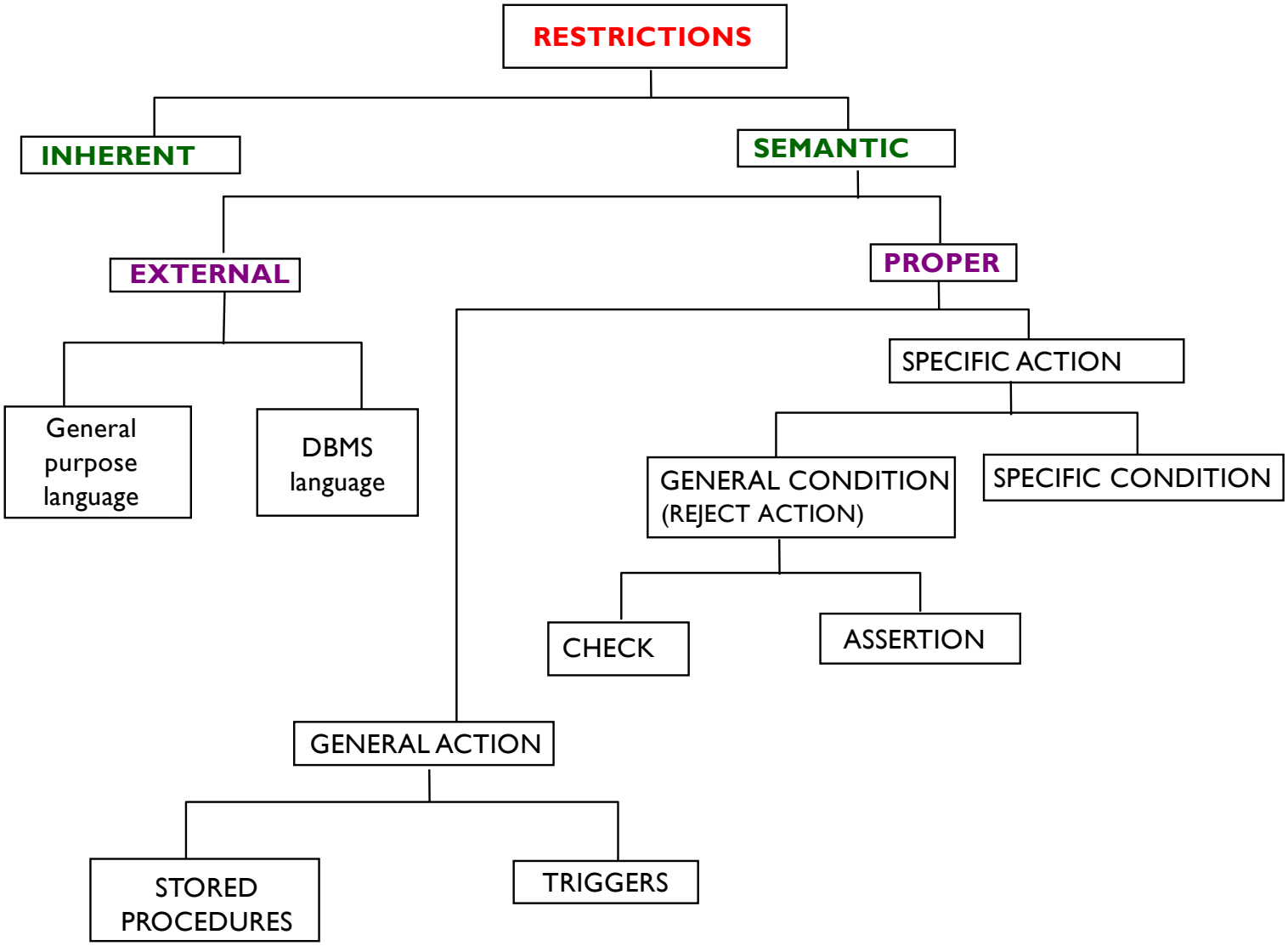


Data semantics 'spread' vs 'integrated'

Integrity restrictions
Components

- An integrity restriction is formed by:
  - Modifying ***operation*** (insert, delete or update) whose execution triggers the restriction checking.
  - ***Condition*** to be accomplished. It is a logical proposition, defined on one or several schema elements, which can be true or false.
  - ***Action*** to be performed depending on the evaluation result of the condition.
- Integrity restriction can be considered as **ECA rules** (**Event, Condition, Action**):
  - when an event happens, a condition is checked and, depending on the checking result, an action is triggered (deny an operation, show a message, correct an error, etc.).
- Besides these elements, they can also have a ***name***, so they can be identified, and they may also indicate the ***time*** when the condition must be assessed

Integrity restrictions
Classification



- **Own restrictions** are specified by defining the schema using the facilities provided by the data definition function. They are stored in the database (not in programs), so they cannot be violated by any application, i.e., any update must respect them.
- Depending on the need of defining or not the action, we can find:
  - **General action**: a procedure (in a programming language) must be written for determining the action to be performed. They subdivide into:
    - **Stored procedures**: fully defined in a procedural way (both action and condition).
    - **Triggers**: A condition is formulated (through a logical proposition). If the condition is accomplished, it triggers an action procedurally specified.
  - **Specific action**: the action (normally a denial) is implicit in the restriction itself.

- Several types can be found:
  - **General condition**: the condition is defined through a **logical proposition**. The operation is an update. The action is not declared as there is an implicit denial: the system evaluates the condition, if the result is true, the update is performed, otherwise no action is performed. Two types are included in **SQL**:
    - **Check**: the logical expression used for formulating the condition is defined on one or several attributes of a single element. E.g., a **CHECK** clause in a CREATE TABLE sentence.
    - **Assertion:** like checks, but they can refer to more than one schema element. E.g.: **CREATE ASSERTION**.
  - **Specific condition**: a.k.a. 'special case' or 'implicit'. They refer to several options provided by the DM when defining the schema elements. E.g., in the Relational Model we can find: **PRIMARY KEY**, **FOREIGN KEY**, **NOT NULL** ...

CINEMA (1/2)

- We want to computerize a cinema in Ciudad Real (for storing current and historical information). From the result of the analysis carried out, the following semantic assumptions or business rules have been obtained:

  a) This cinema consists of several rooms, identified by their number. For each room, it is interesting to know its capacity (number of seats), its number of rows, and the floor number on which it is located.

  b) For each room, each seat is assigned a row number and seat number. It must be checked that the row number of a seat is not greater than the number of rows in each room.

  c) For each room and each day, there can be several screenings and each of them from a different movie.

  d) Each movie has an identifier, a title, its length, director/s, genre and cast.

CINEMA (2/2)

e) Each director and cast member has an identifier, name and surname. For each actor/actress we need to know its role in the movie (principal or supporting actor). A director can be an actor and viceversa.

f) We want to record the ticket sales. For each ticket, we want to know the ticket number, date, price, room, movie, screening and the seat location (row and seat number).

g) Each room is assigned an usher for each screening. An usher can work in different rooms on the same day. In each room on the same day, but in a different screening, there may be a different usher. For each usher we need to record its identifier, names and surname.

Chess championship

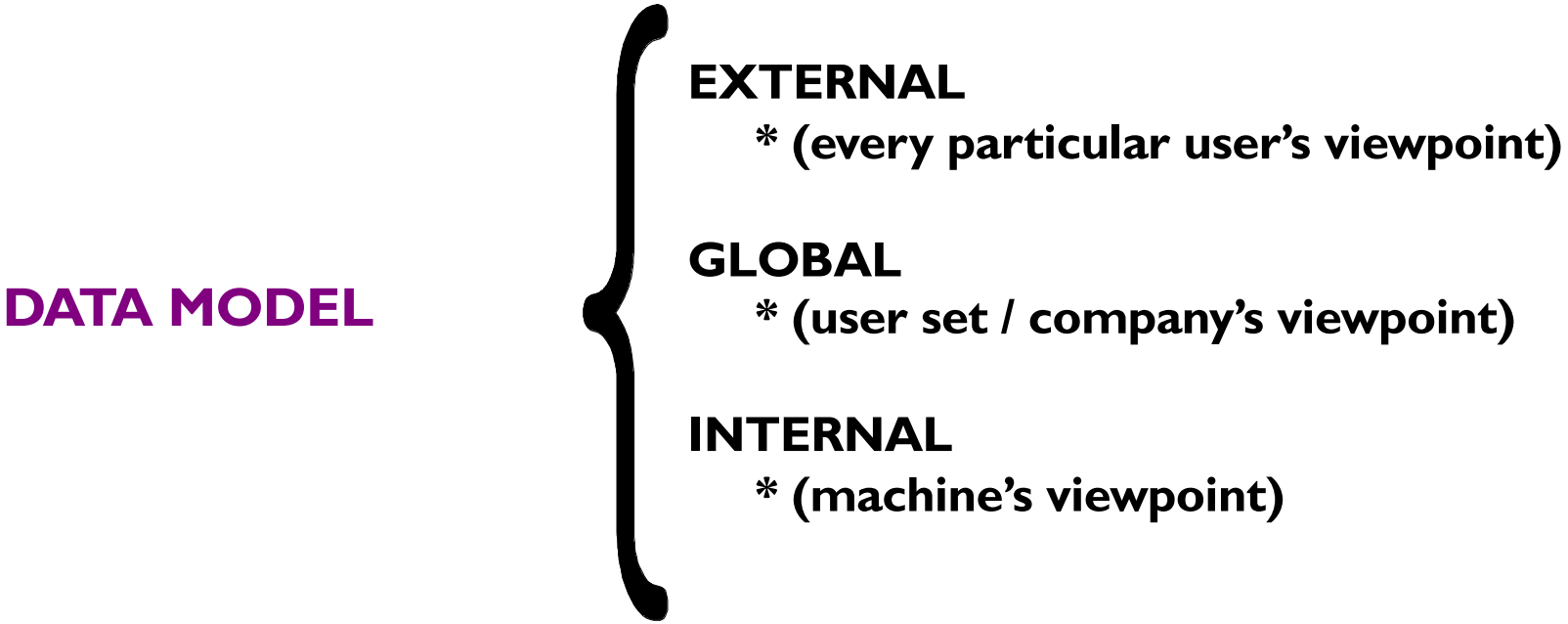
- In this championship, we will find players and referees. We want to know (from all of them) the associate number, their name, address, telephone and number of previous championships (as player and/or referee). Besides, game level (1-10 scale) will be needed for players.
- Referees cannot be players and vice versa.
- Every country can send a set of players and referees, although not every country in the world will do it. Every player and referee will be sent by a single country. A country can be represented by another country.
- A country is identified by a correlative number (alphabetical order). We also need to know its name and the number of chess clubs in that country.
- Every game is identified by a correlative number (id_G). It is played by two players and a referee is in charge of looking after the chess rules to be accomplished. We need to know each player's band (white or black). A referee cannot judge a game if a player from his/her same country would play it.
- Every player plays, at least, one game.

Chess championship

- Players and referees will be hosted in a hotel. We need to know which hotel is every person is hosted at and arrival & departure dates. We also need to know every hotel's name (which identifies it), its address and telephone number.
- The championships takes place during a set of days (year+month+day) and every game takes place in one of these days. There can be days without games.
- Every game is played in a room (identified by id_R). Each room is in a hotel and we need to know how many tickets have been sold to watch a game live in the corresponding room. We also want to know each room's capacity and the devices in it (TV, radio, DVD, etc.).
- Every movement in a game must also be recorded. It will be done using a number indicating the movement order within the game and the play (origin and destination positions) and a brief comment from an expert.

DM Taxonomy

- According to the ANSI architecture, we can find:



- External and global DM are also called **logical**, as they refer to logical aspects of data, while internal DM refer to **physical** aspects.

GLOBAL DMs

CONCEPTUAL OR SEMANTIC

- Focused on describing the real world independently of the machine -

- Entity/Relationship (E/R)
- Object Orientation (UML)

CONVENTIONAL OR LOGICAL

- Oriented towards a DBMS implementation-

- Hierarchical
- Network (Codasyl)
- Relational

CONVENTIONAL

- Implemented in commercial DBMSs
- DBMS dependant
- Closer to computer
- Little semantic capabilities
- Focused towards implementation
- Computer scientist/system interface
- *Mediation* level between external and internal levels

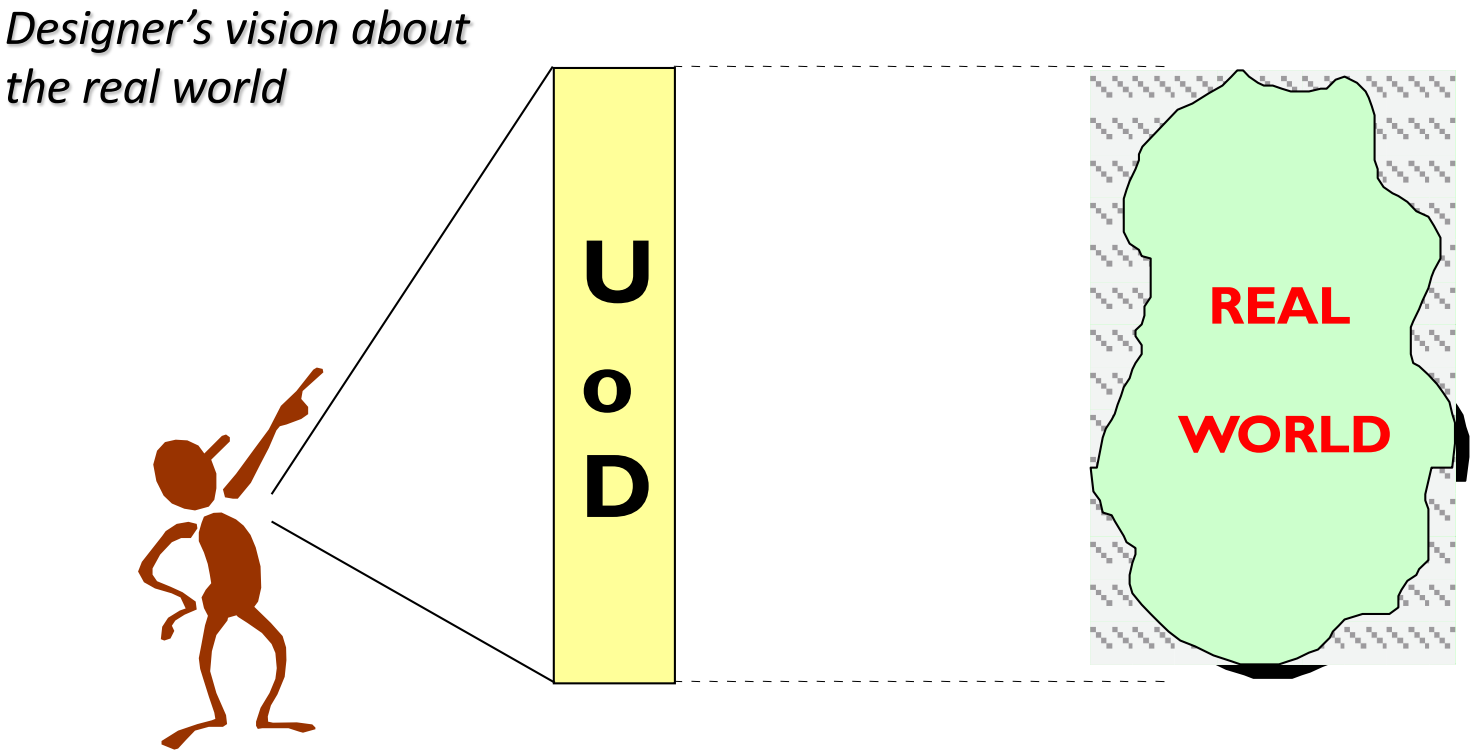
CONCEPTUAL

- Not implemented in DBMSs
- DBMS independent
- Higher abstraction level
- More semantic capabilities
- Focused in high level design (conceptual modelling)
- User/computer scientist interface

- A DB design process is the set of tasks needed to go from a certain reality (UoD) to the DB that represents that reality. DMs play an important role on a DB design process as they provide a set of abstraction facilities which help us represent the reality.

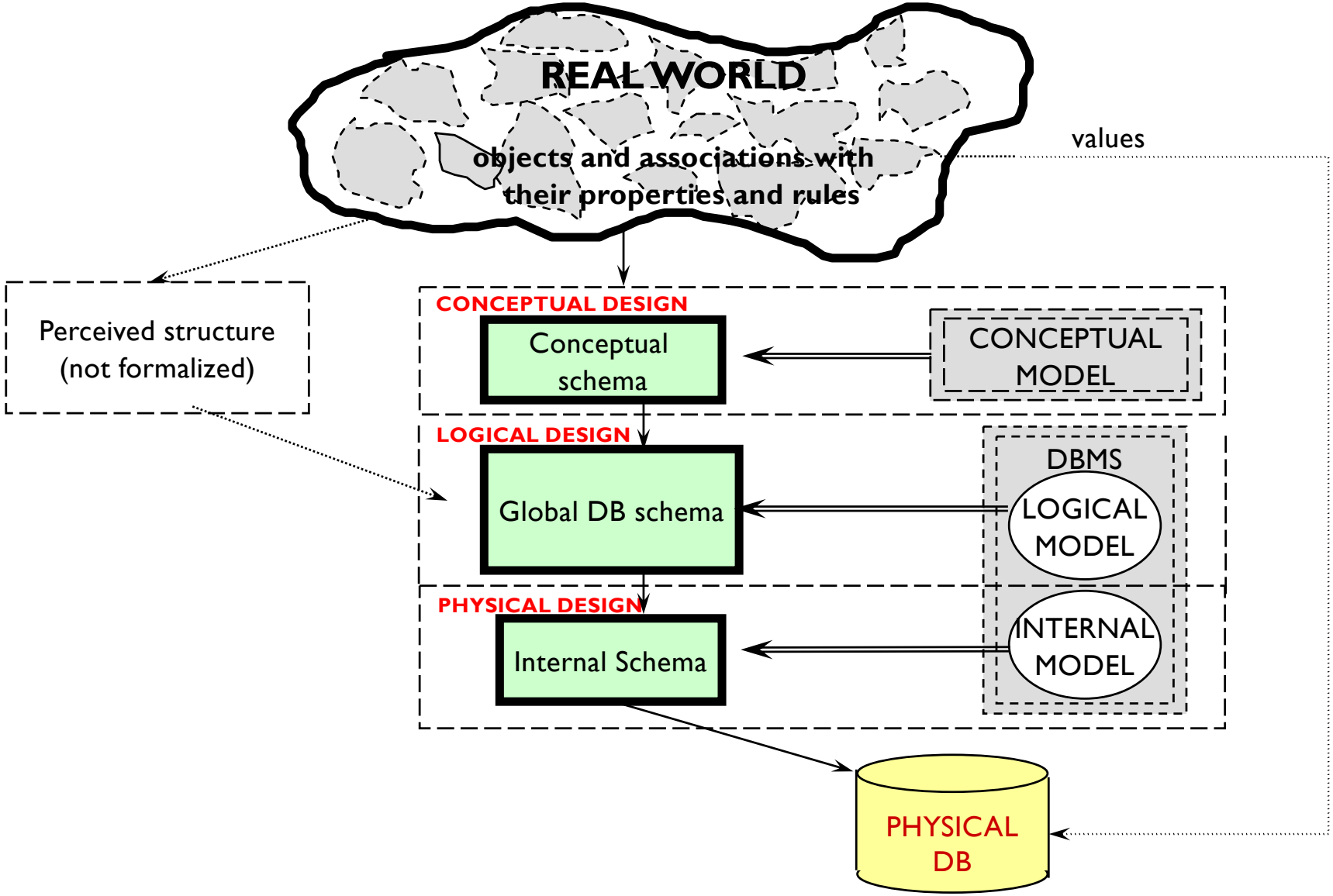
- The **goals** of every DM can be:
  - a) **Formalization**: the DM allows the formal definition of the allowed structures and restrictions; it also establishes the base for the definition of a data language.

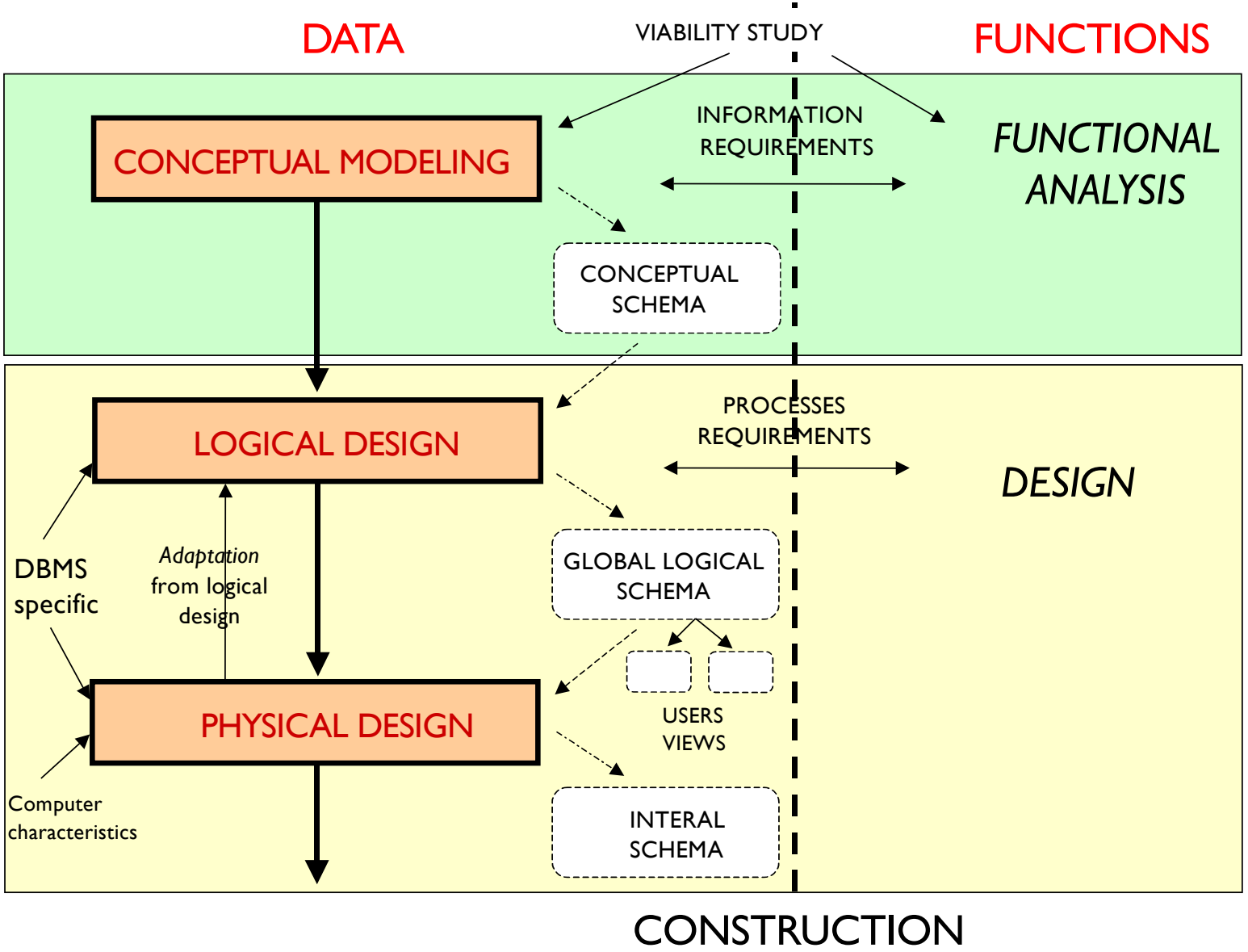
  - b) **Design**: the DM is a key element in the development of a DB design methodology, and the other components of the methodology (languages, documentation, and other tools) are based on it.



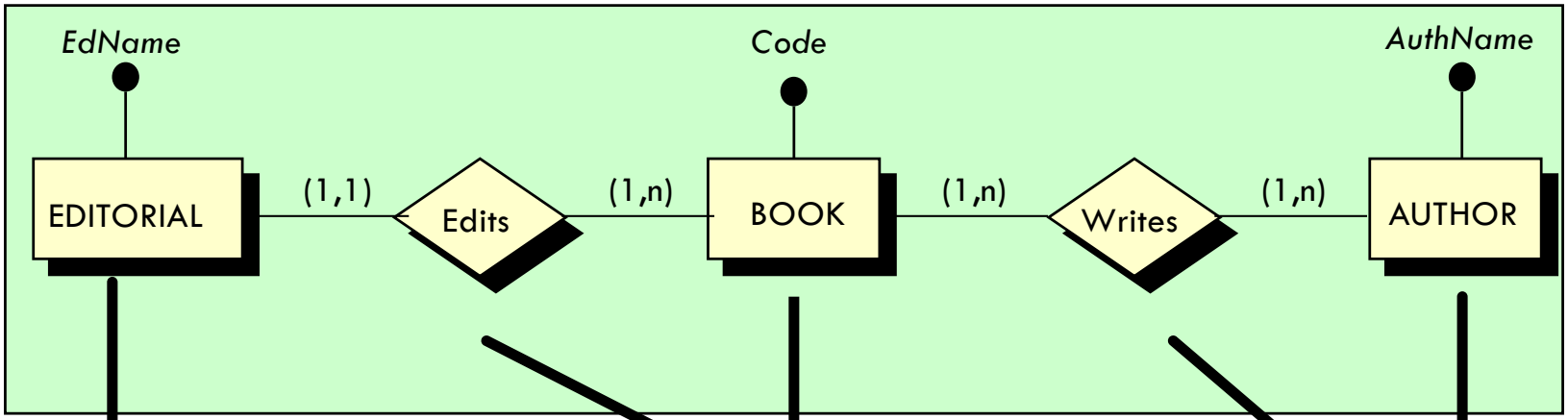
Universe of Discourse vs Real World

DM & DB design


REAL WORLD
objects and associations with their properties and rules
values
Perceived structure (not formalized)
CONCEPTUAL DESIGN
Conceptual schema
CONCEPTUAL MODEL
LOGICAL DESIGN
Global DB schema
DBMS
LOGICAL MODEL
PHYSICAL DESIGN
Internal Schema
INTERNAL MODEL
PHYSICAL DB


DATA
VIABILITY STUDY
FUNCTIONS
CONCEPTUAL MODELING
INFORMATION REQUIREMENTS
FUNCTIONAL ANALYSIS
CONCEPTUAL SCHEMA
LOGICAL DESIGN
PROCESSES REQUIREMENTS
DESIGN
DBMS specific
Adaptation from logical design
GLOBAL LOGICAL SCHEMA
PHYSICAL DESIGN
USERS VIEWS
Computer characteristics
INTERAL SCHEMA
CONSTRUCTION

- **Stage 1:**
  - Conceptual design: the goal is obtaining a good representation of the information resources, independently of the applications and equipment (DBMS), and without any concern about efficiency.
- **Stage 2:**
  - Logical design: the goal is transforming the conceptual model by adapting it to the data model which supports the DBMS to be used.
- **Stage 3:**
  - Physical design: the goal is achieving an implementation as efficient as possible of the logical schema.



ER Model

BOOK(<u>Code</u>, *Title, Language, ..., Editorial*)

Foreign Key

EDITORIAL (<u>*EdName*</u>, *Address, City, Country*)

Foreign Key

WRITES(<u>*Author, Book*</u>)

Foreign Key

AUTHOR (<u>*AuthName*</u>, *Nationality, Institution*)

Relational Model

- The last stage in the DB design methodology is **physical design**, intended to **satisfy the system requirements for optimizing the cost/profit relation**.

- This is achieved by:
  - Decreasing waiting time
  - Minimizing storage space
  - Avoiding periodical rearrangements
  - Providing maximum security
  - Optimizing resources consumption

- The **inputs** of the physical design stage are:
  - A list with the goals of the physical design with priorities and quantification (if possible);
  - Specific logical design;
  - Available hardware resources;
  - Available software resources (operating system, middleware, ...);
  - Information about the applications that the DB will use;
  - Data security policies.
- The **outputs** produced will be:
  - Internal structure (schema);
  - Specifications for DB *tuning;*
  - Security rules.

There is not a formal, generic model for physical design; it depends on each specific commercial product.