Unit 4:
Relational Database Design

4.1. Database Design Fundamentals

4.2. Conceptual Design

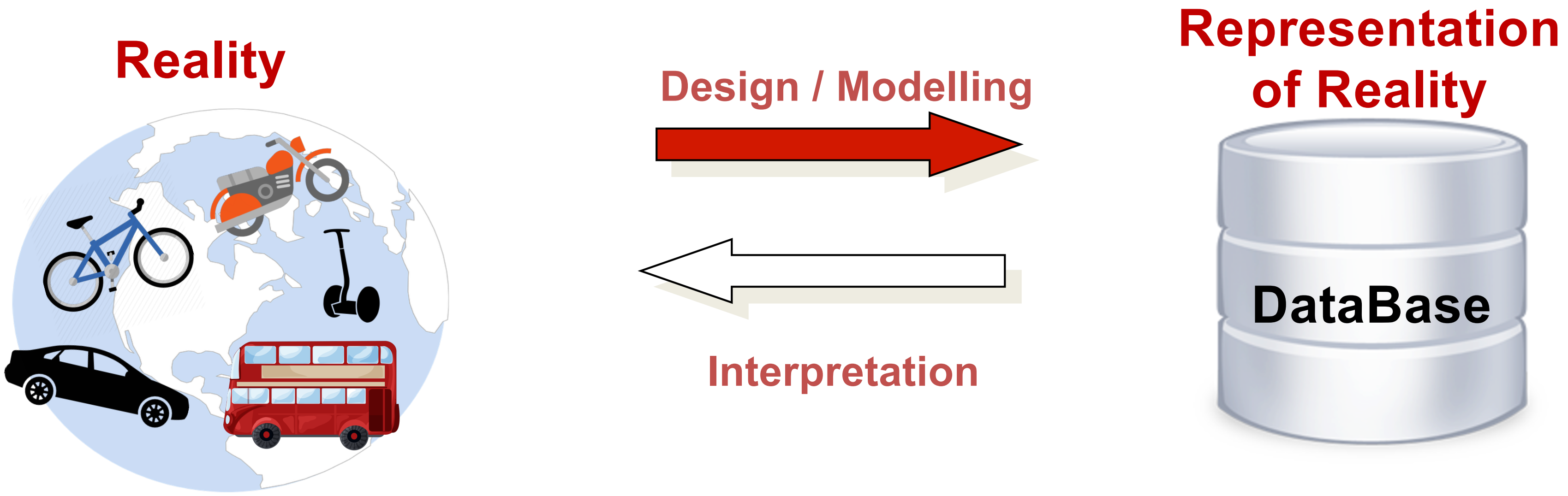
4.3. Logical Design

Unit 4.1 DB Design Fundamentals

1. **Introduction**
2. Methodology
3. Data models
4. Database Design
5. Example

1. Introduction

In this unit we will present a methodology for the design of relational databases



Reality

Design / Modelling

Interpretation

Representation of Reality

DataBase

We will focus on:

- Methodology issues: Strategies and recommendations to address the design problem.
- Modelling languages issues: Presentation of an appropriate (graphical) language to represent the system (data model).

UD 4.1 DB Design Fundamentals

1. Introduction
2. **Methodology**
3. Data models
4. Database Design
5. Example

2. Methodology

A methodology is a set of standard procedures, techniques and documentation for the development of a product (a database in our case).

A methodology is supported by:

- Techniques: how to deal with each of the **steps** and **activities** in the methodology
- Models: Way to **represent** or think abstractly about the reality, the problem or the solution.
- CASE tools: (optionally) **software** tools to automate or assist on the development of techniques and models.

2. Methodology

Analysis:

Obtain the set of requirements of information and of process that the organization needs for achieve its aims.

Design:

Conceptual Design:

Obtain a representation of reality including static and dynamics properties necessaries to meet the organization's requirements.

Logical Design

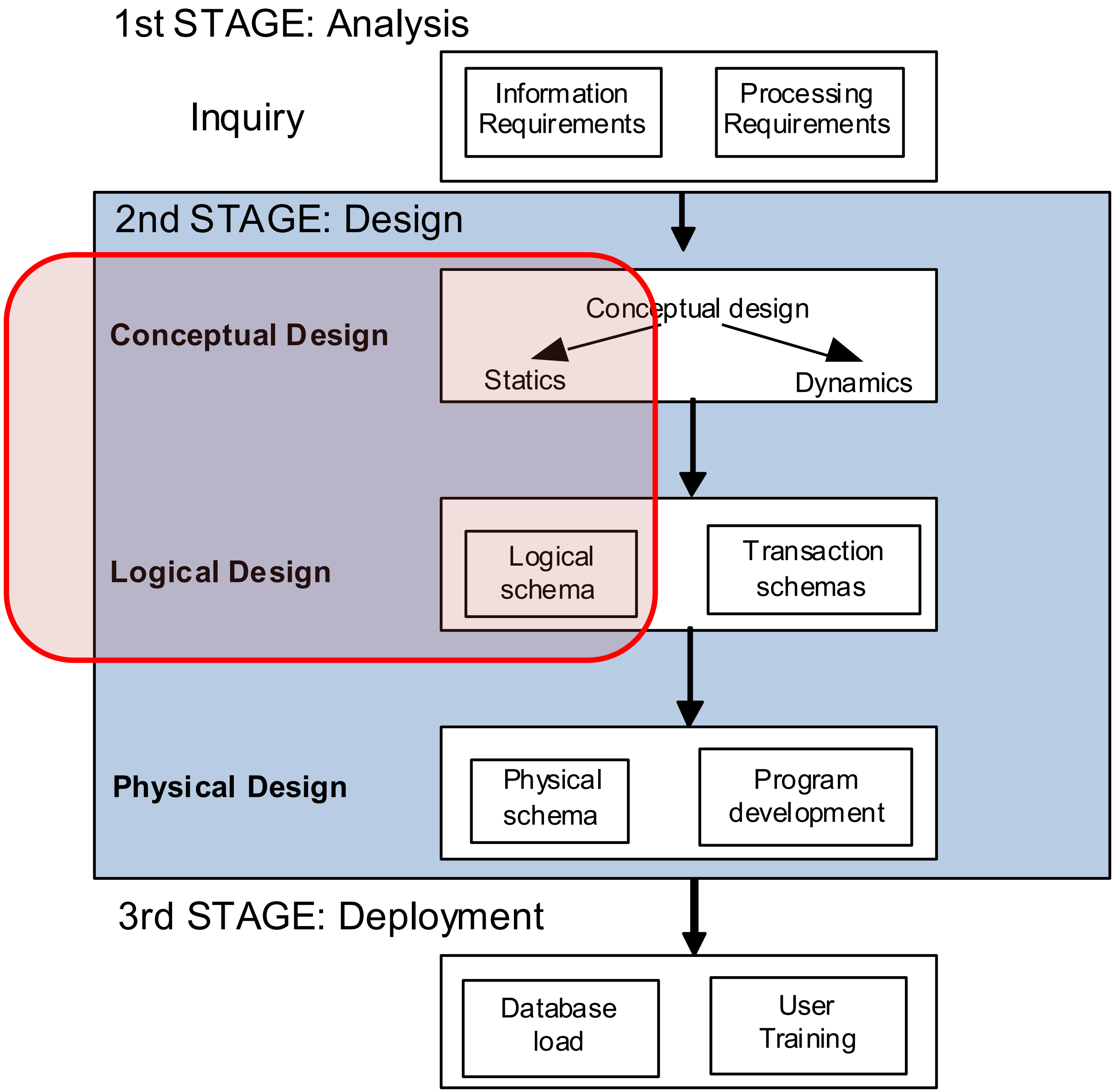
Translation of the conceptual scheme to the data model of the DBMS to use.

Physical Design

Selection of storage structures taking into account details of the physical representation for obtaining a good performance.

Deployment:

Incorporation of the database and applications into the organization.


1st STAGE: Analysis
Inquiry
Information Requirements
Processing Requirements
2nd STAGE: Design
Conceptual Design
Conceptual design
Statics
Dynamics
Logical Design
Logical schema
Transaction schemas
Physical Design
Physical schema
Program development
3rd STAGE: Deployment
Database load
User Training

UD 4.1 DB Design Fundamentals

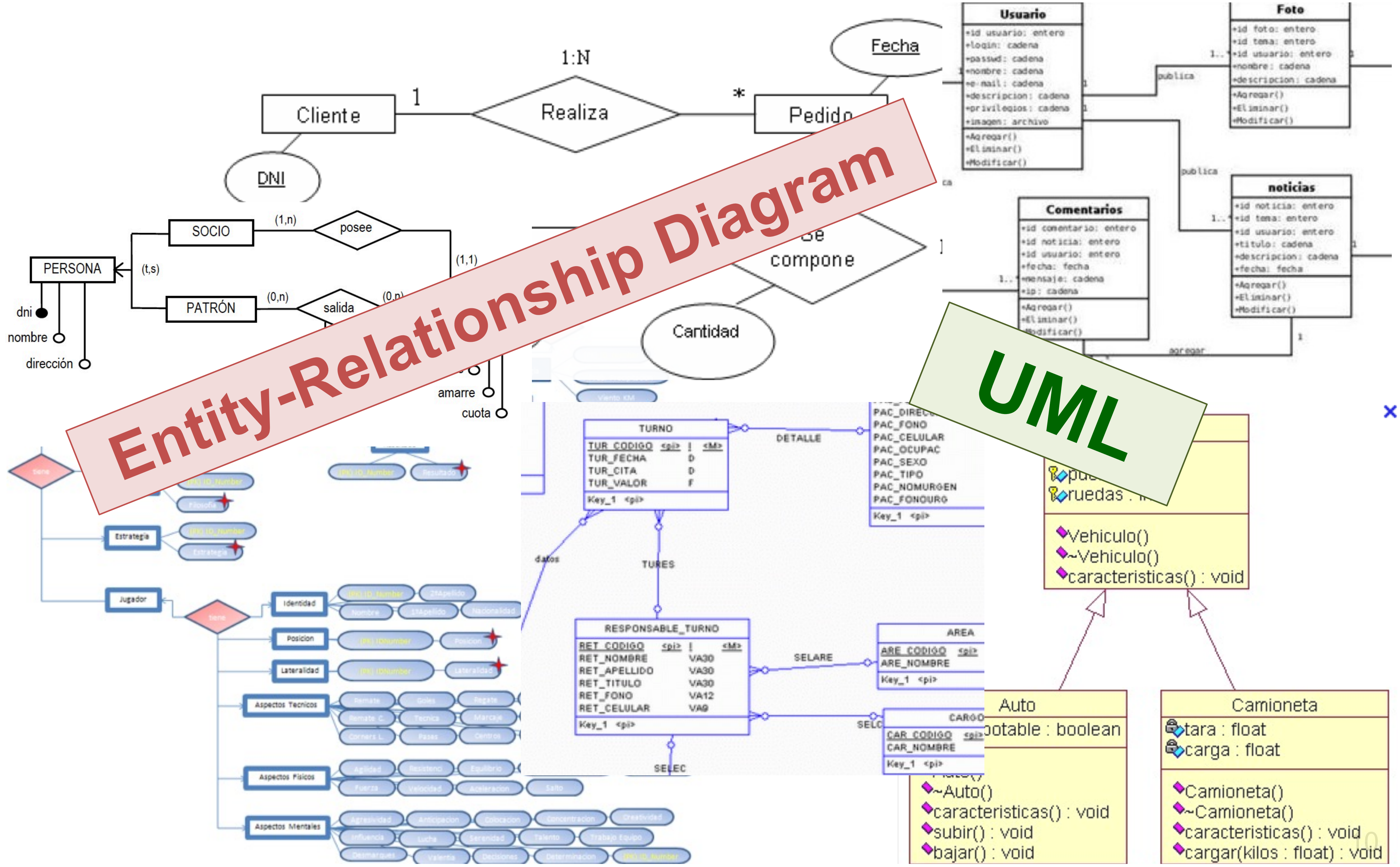
1. Introduction
2. Methodology
3. **Data models**
4. Database Design
5. Example

3. Data models

A **data model** is a way in which static and dynamic properties of reality (entities, relationships,…) are represented.

Example: The relational data model

There are many models, many notations,…



Entity-Relationship Diagram

UML

We are going to use a conceptual data model that:

- incorporates notions from the Entity-Relationship Model using UML
- is more abstract , expressive, and system-independent than the classical relational model
- is essentially graphical (based on UML)

UD 4.1 DB Design Fundamentals

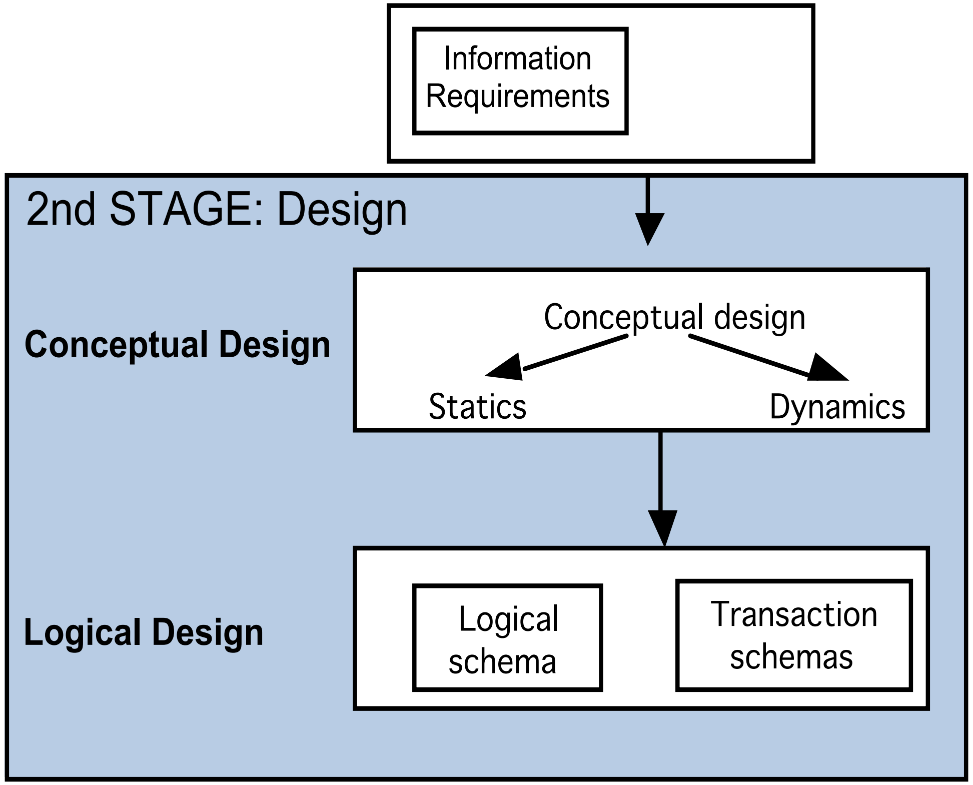
1. Introduction
2. Methodology
3. Data models
4. **Database Design**
5. Example

4. Database Design

UML Classes Diagram (Unit 4.2)

Logical Design
(transformation)

Relational Model (Unit 4.3)



UD 4.1 DB Design Fundamentals

1. Introduction
2. Methodology
3. Data models
4. Database Design
5. **Example**

1. Analysis stage: information requirements

Lecturer:

- code, name and address
- department where the lecturer belongs to
- subjects s/he teaches, and how many hours each
- total number of teaching hours s/he is assigned

Department:

- name, head and telephone.

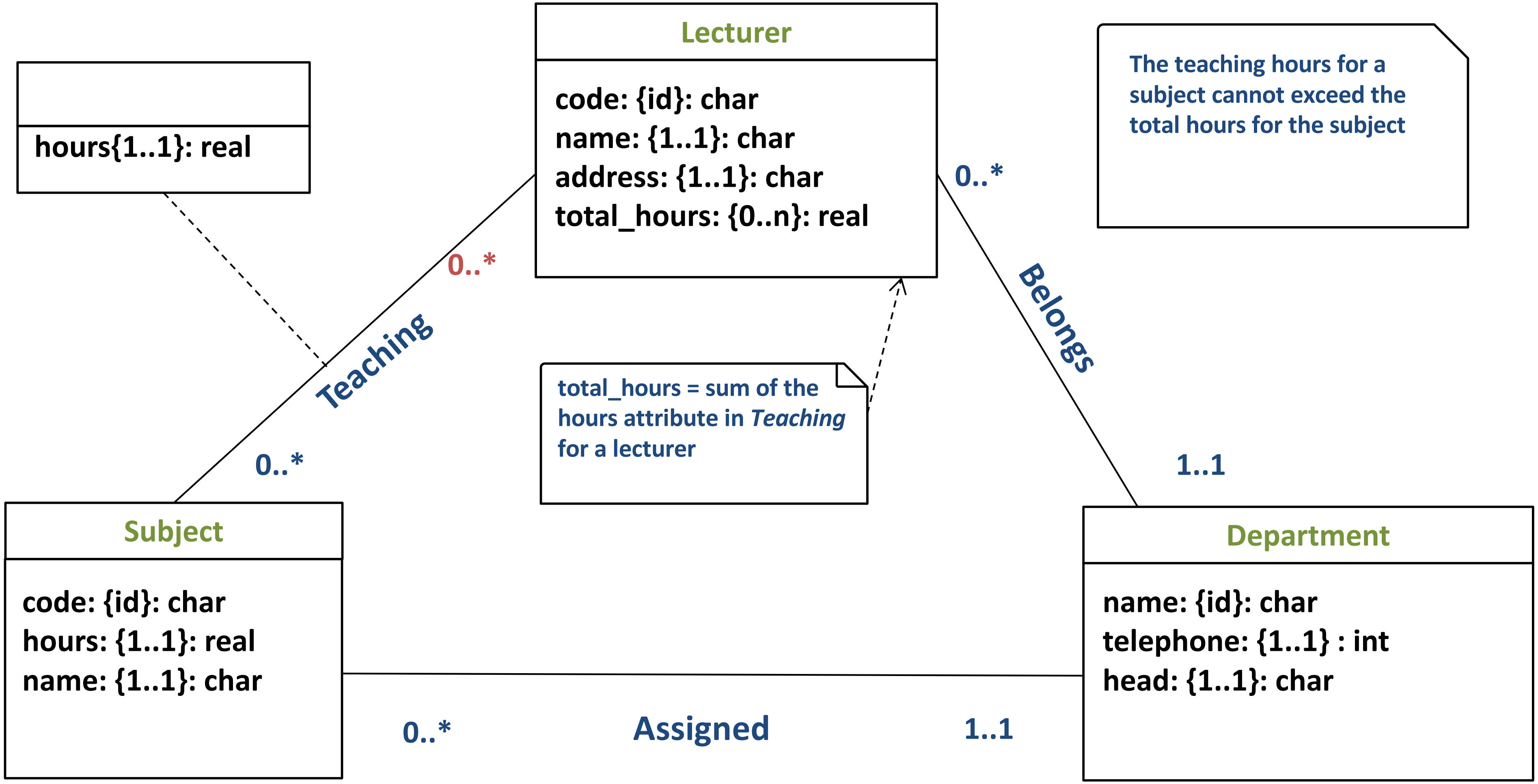
Subject:

- code and name of the subject,
- total number of hours in the degree syllabus
- department which is assigned to.

INTEGRITY CONSTRAINTS:

- A lecturer must belong to one and only one department.
- A subject must be assigned to one and only one department.
- There can't be two departments with the same name.
- There can't be two lecturers with the same code.
- There can't be two subjects with the same code.
- The number of taught hours for a subject cannot exceed the total number of hours for the subject in the degree syllabus.
- A lecturer cannot teach more than 12 hours in one subject.
- A subject cannot have more than 24 hours.

2a. Design stage: Conceptual Design (static)



The teaching hours for a subject cannot exceed the total hours for the subject

total_hours = sum of the hours attribute in *Teaching* for a lecturer

UML Classes Diagram

2a. Design stage: Conceptual Design (dynamic)

Transaction *Insert_lecturer*

Insert into Lecturer

Insert into Belongs

Transaction *Insert_subject*

Insert into Subject

Insert into Assigned

Transaction *Insert_departament*

Insert into Departament

…

Transaction description

2b. Design stage: Logical Design (static)

Department (name: char(50), head: char(50), telephone: char(8))

PK: {name}

Lecturer (code: char(9), name: char(50), address: char(50), dname: char(50))

PK: {code}

FK: {dname} → Department

NNV: {name, address, dname}

Subject (code: char(5), name: char(50), hours: number, dname: char(50))

PK: {code}

FK: {dname} → Department

NNV: {name, hours, dname}

Teaching (lcode: char(9), scode: char(5), hours: number)

PK: {lcode, scode}

FK: {lcode} → Lecturer

FK: {scode} → Subject

NNV: {hours}

*(*) The attribute **total_hours** is not included and will be calculated every time it is needed.*

(**) The number of taught hours for a subject cannot exceed the total number of hours for the subject in the degree syllabus.

2b. Design stage: Logical Design (dynamic)

TRANSACTION Insert_lecturer (code: char(9), name: char(50), address: char(50), dname: char(50))

INSERT INTO Lecturer **VALUES** (code, name, address, dname)

TRANSACTION Insert_subject (code: char(5), name: char(50), hours: number, dname: char(50))

INSERT INTO Subject **VALUES** (code, name, hours, dname)

TRANSACTION Insert_departament (name: char(50), head: char(50), telephone: char(8))

INSERT INTO Department **VALUES** (name, head, telephone)

2c. Design stage: Physical design

Lecturer:

File indexed by code; index on name

Subject:

File indexed by code; index on name

Department:

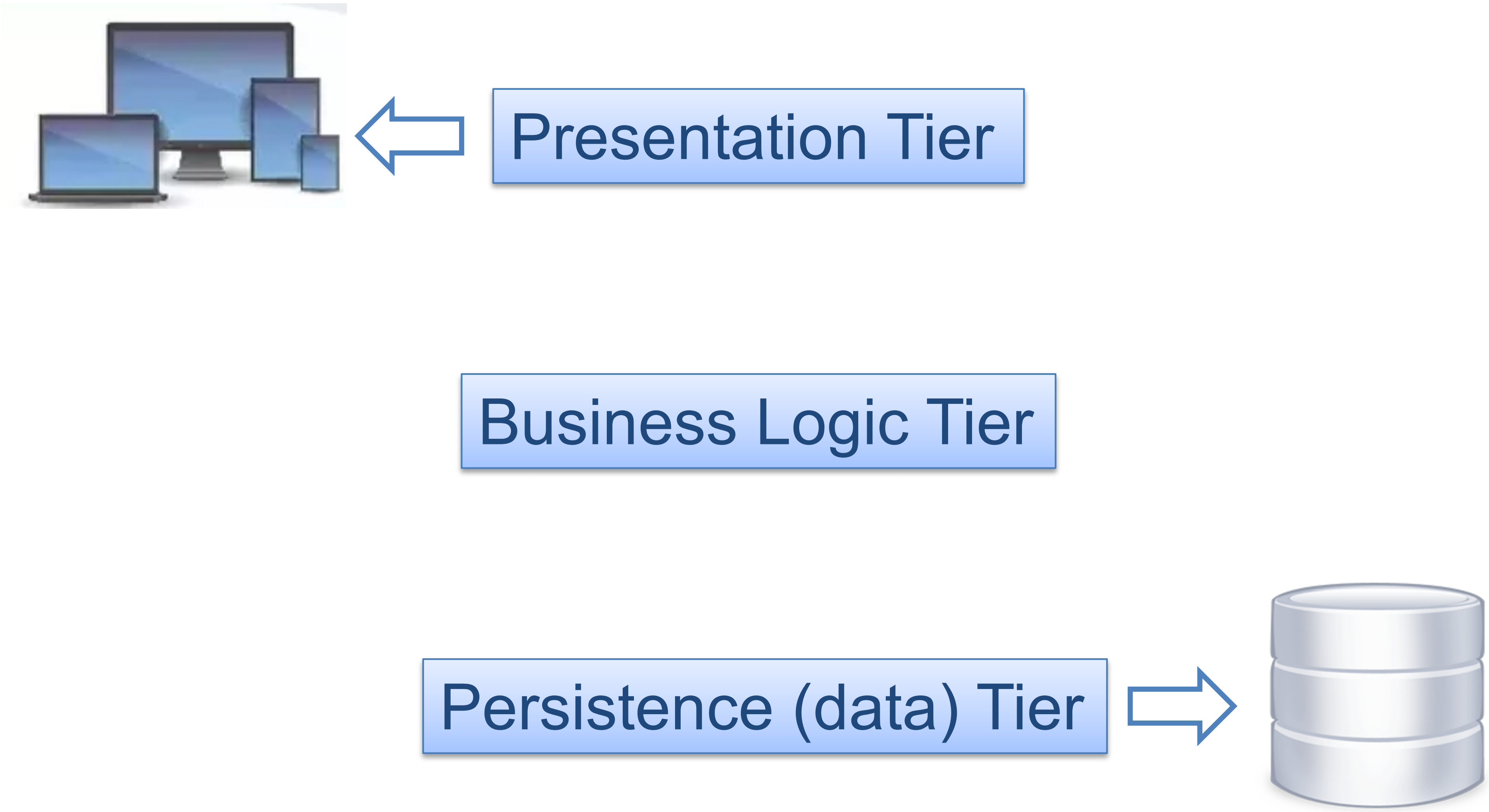
Sequential file; index on name

Teaching:

File indexed by scode; index on lcode

Databases vs Software Engineering

3-Tier Architecture



Presentation Tier

Business Logic Tier

Persistence (data) Tier