Unit 1:
Relational Databases

1.1. Fundamentals

1.2. The Relational Data Model

1.3. Interpretation of a Relational Database

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

4 Constraints and transactions

1 Introduction

Historical milestones about the relational data model

70’s: Proposed by E. Codd in 1970

80’s: Becomes popular in practice (Oracle, ...). ANSI defines the SQL standard.

90’s: Generalization and standardization (SQL’92) and extensions.

Reasons of success:

Simplicity: a database is a “set of tables”.

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

2.1 Informal view of a relational database

2.2 Relational database goals

3 The relational data model

4 Constraints and transactions

2.1 Informal view of a relational database

The information is organized in tables, with columns and rows:

- Entities are represented as tables (a.k.a. relations).
- Objects (entity instances) correspond to table rows (or tuples).
- Object's features are represented by attributes. These attributes correspond to the columns of the tables, and are also known as fields.
- Attributes in the same column must have the same datatype (domain).

2.1 Informal view of a relational database

Teaching:

Lecturer code *(cod_pro)*
Subject code *(cod_asg)*
Lecture groups *(GT)*
Labs groups *(GP)*

Lecturer:

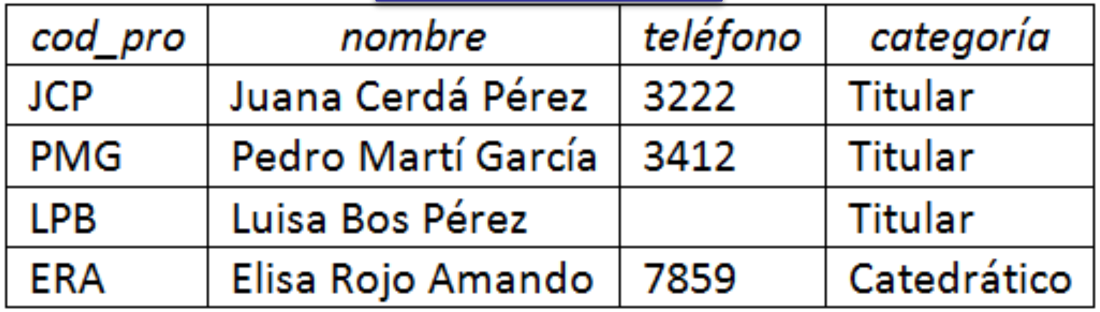
Code *(cod_pro)*
Name *(nombre)*
Telephone *(telefono)*
Category *(categoría)*

Subject:

Code *(cod_asg)*
Name *(nombre)*
semester in which is offered *(semester)*
lecture credits *(T)*
lab credits *(P)*

2.1 Informal view of a relational database

Lecturer



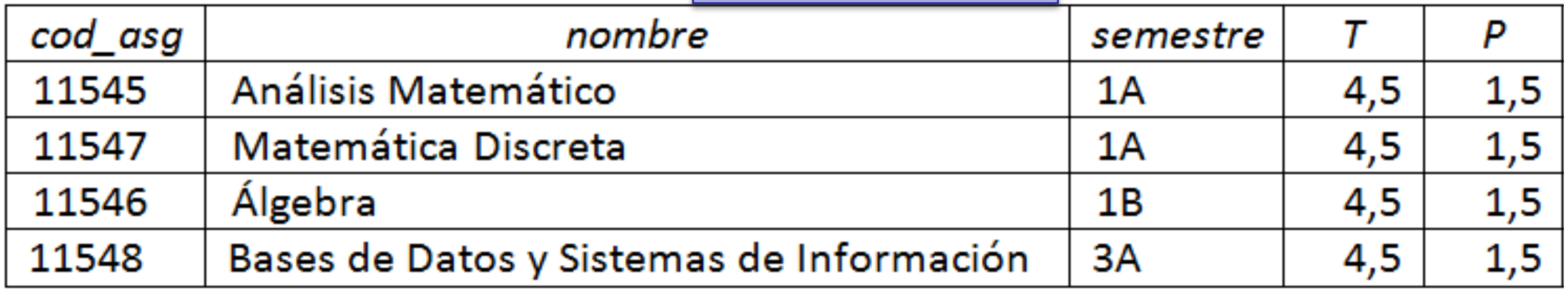
cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Row (*Tuple*) = lecturer instances
Column (*Attribute*) = property → with a ***name*** and an associated ***type***

LECTURER (cod_pro:char(5), nombre:char(40), telefono:char(9),
categoria:char(30))

2.1 Informal view of a relational database

Subject



cod_asg	*nombre*	*semestre*	*T*	*P*
11545	Análisis Matemático	1A	4,5	1,5
11547	Matemática Discreta	1A	4,5	1,5
11546	Álgebra	1B	4,5	1,5
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5

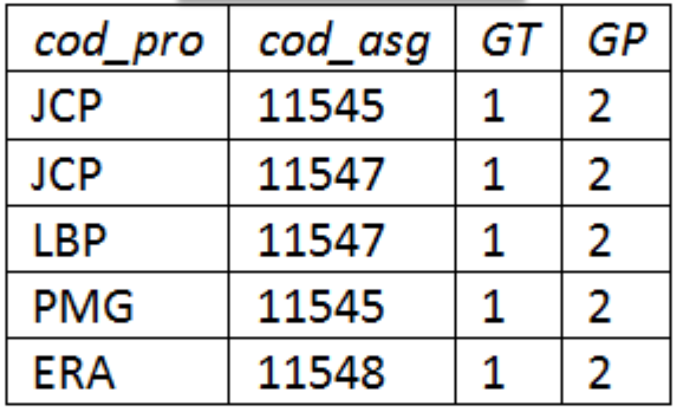
Row (*Tuple*) = subject instances

Column (*Attribute*) = property → with a ***name*** and an associated ***type***

SUBJECT (cod_asg:char(5), nombre:char(40), semester:char(2),
T:integer, P:integer)

2.1 Informal view of a relational database

Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

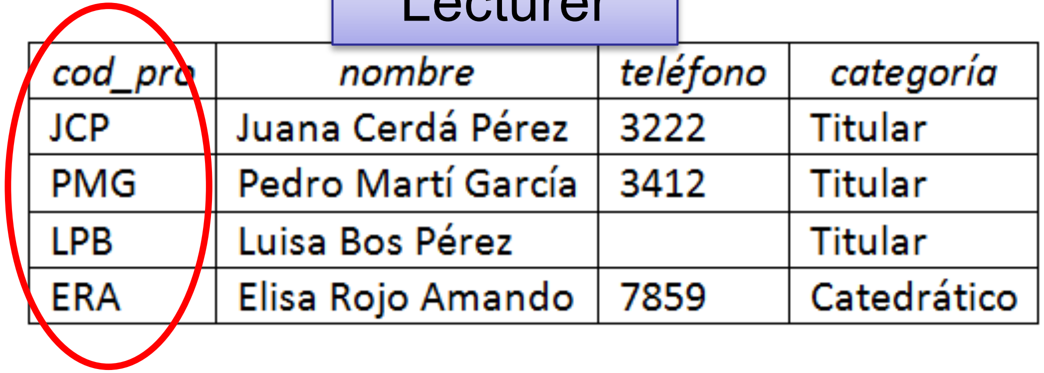
Row (*Tuple*) = teaching instances

Column (*Attribute*) = property → with a ***name*** and an associated ***type***

TEACHING (cod_pro:char(5), cod_asg:char(5), GT:integer, GP:integer)

2.1 Informal view of a relational database

Lecturer

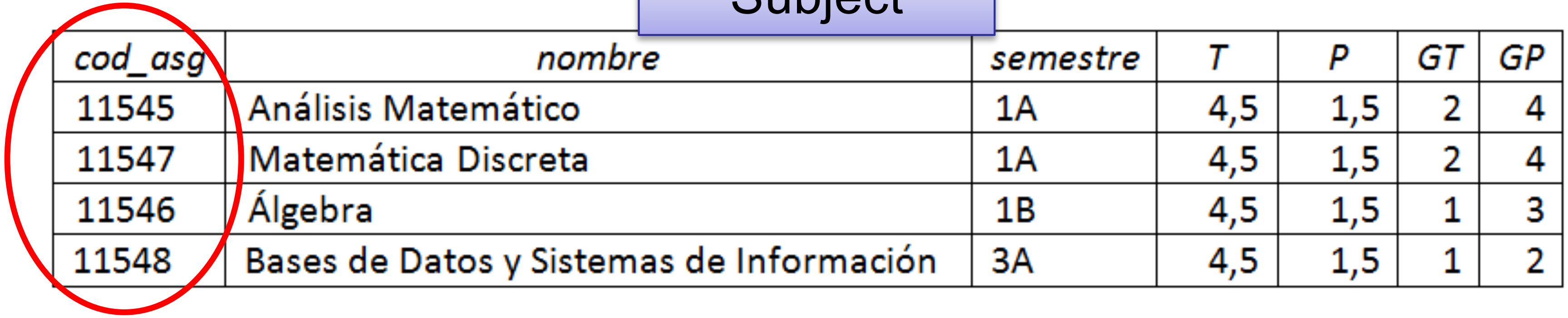


cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

There are attributes which identify the tuples of a relation:

cod_pro in LECTURER,
cod_asg in SUBJECT.

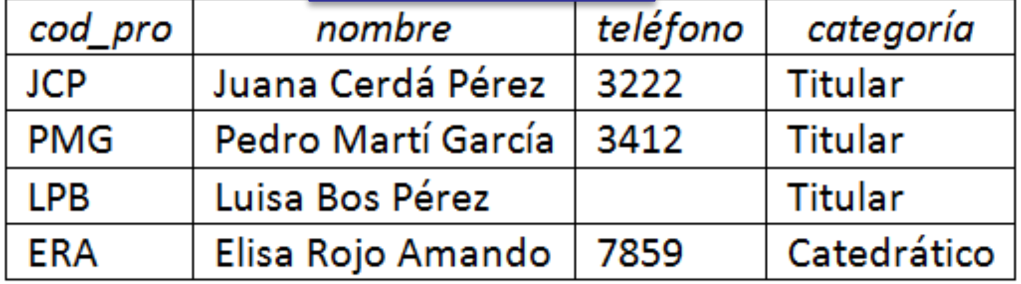
Subject



cod_asg	*nombre*	*semestre*	*T*	*P*	*GT*	*GP*
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5	1	2

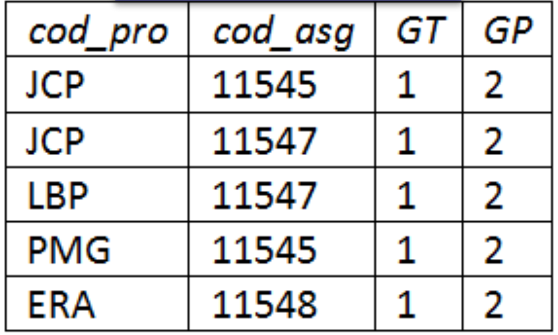
2.1 Informal view of a relational database

Lecturer



cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Teaching

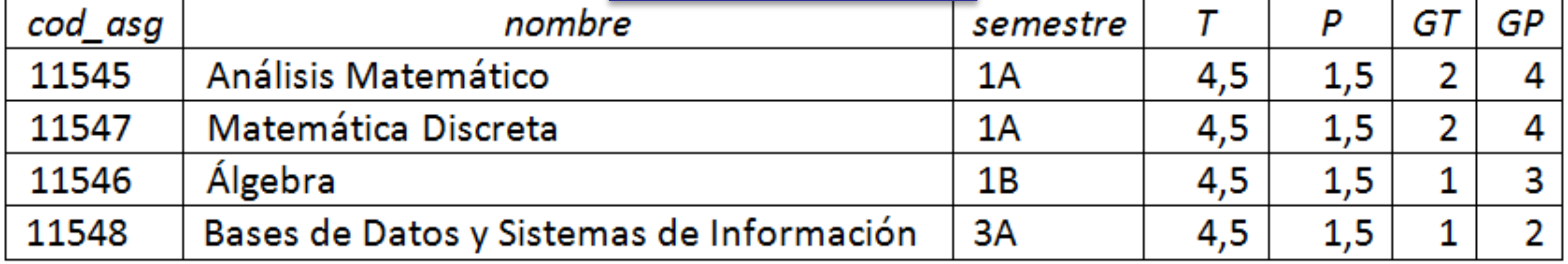


cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

There are attributes which associate two relations:

cod_pro in TEACHING which associates the teaching arrangement with the lecturer (cod_pro) and the subject (cod_asg)

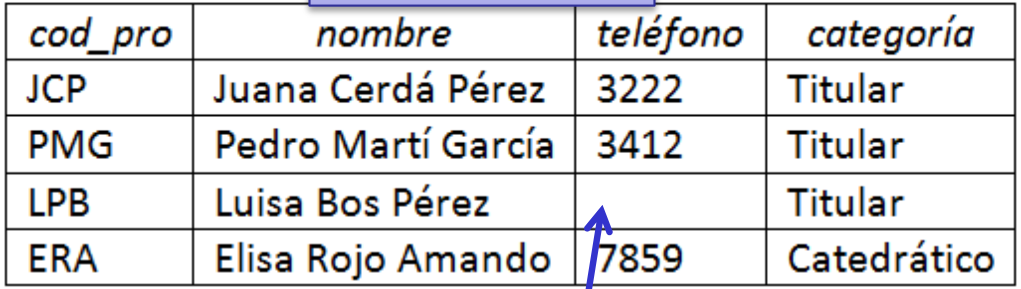
Subject



cod_asg	*nombre*	*semestre*	*T*	*P*	*GT*	*GP*
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5	1	2

2.1 Informal view of a relational database

Lecturer

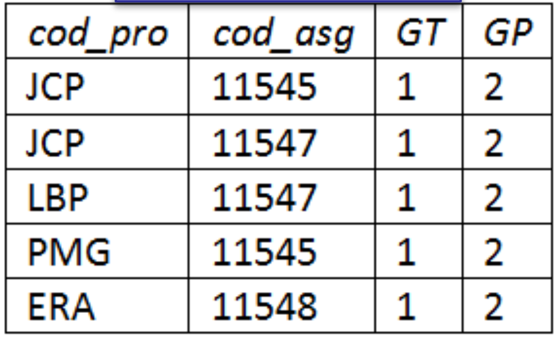


cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

¿?

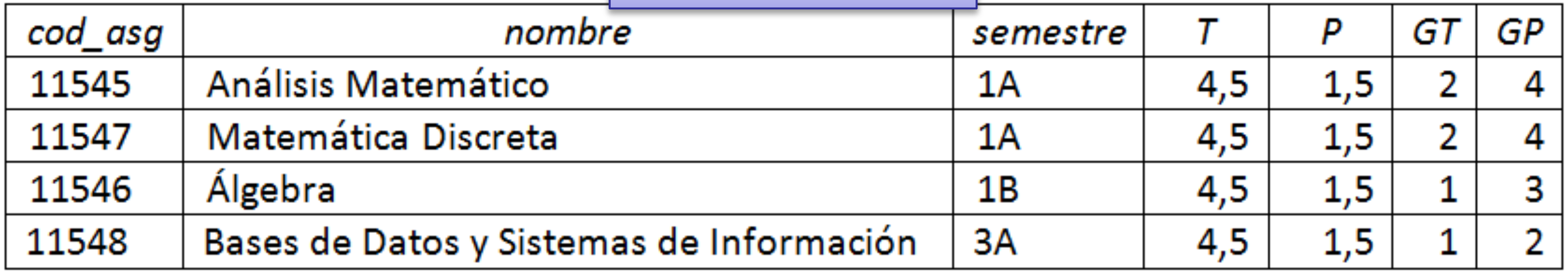
What is the telephone number of “LPB” ?

Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

Subject



cod_asg	*nombre*	*semestre*	*T*	*P*	*GT*	*GP*
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5	1	2

Unit 1.2 The Relational Data Model

1 Introductions

2 Introduction to relational databases

- 2.1 Informal view of a relational database
- 2.2 Relational database goals

3 The relational data model

4 Constraints and transactions

2.2 Relational database goals

The ultimate goal of a database is that users and applications can:

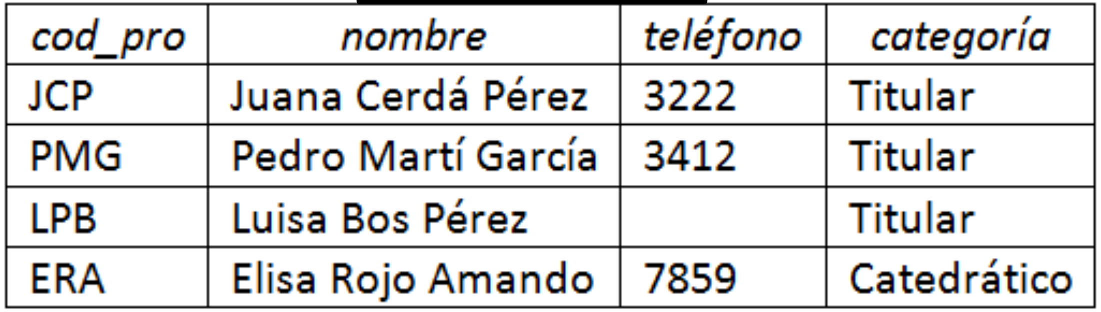
1. Store and modify the information of interest
   - INSERTION
   - DELETION
   - UPDATE

2. Access and retrieve that information:
   - QUERY

2.2 Relational database goals: Modify

- Add a new lecturer: INSERT a row

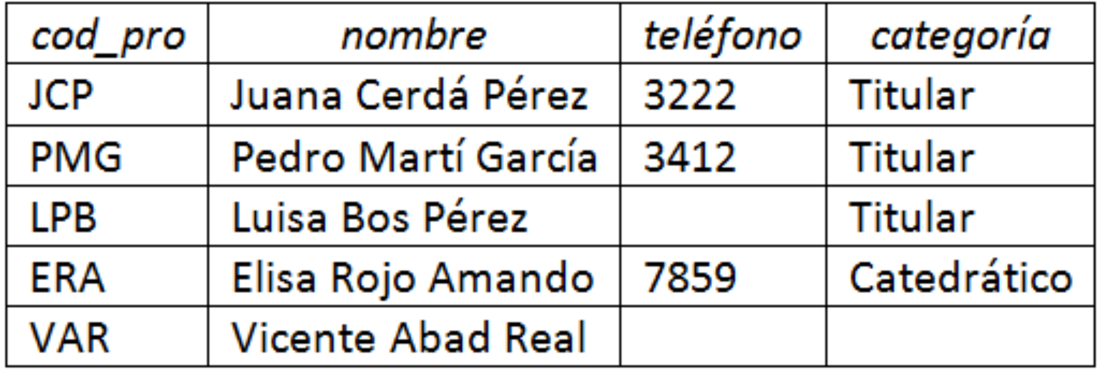
Lecturer



cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Insert a new row:
cod_pro='VAR'
nombre='Vicente Abad Real'

Lecturer

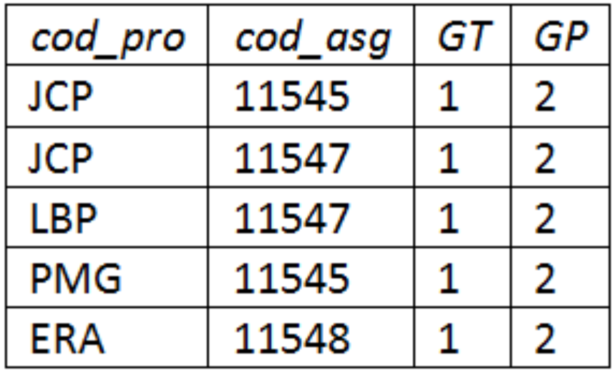


cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático
VAR	Vicente Abad Real		

2.2 Relational database goals: Modify

- Remove the groups of a lecturer: DELETE rows

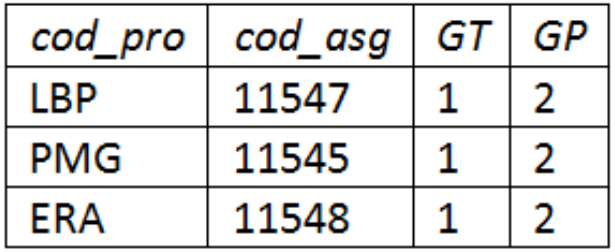
Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

Delete rows where :
cod_pro='JCP'

Teaching



cod_pro	*cod_asg*	*GT*	*GP*
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

2.2 Relational database goals: Modify

- Modify the information of a subject: UPDATE rows

Subject

cod_asg	nombre	semestre	T	P	GT	GP
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5	1	2

Change the row where:

cod_asg=11548

using:

nombre='Bases de Datos Relacionales'

Subject

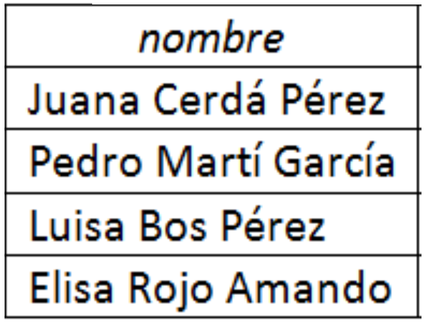
cod_asg	nombre	semestre	T	P	GT	GP
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos Relacionales	3A	4,5	1,5	1	2

2.2 Relational database goals: Queries

A relational query is a retrieval operation to a database which returns part of the information of the database, possibly combined and/or aggregated, in the form of a single table.

Example:

“Obtain the name of all the lecturers”



nombre
Juana Cerdá Pérez
Pedro Martí García
Luisa Bos Pérez
Elisa Rojo Amando

2.2 Relational database goals: Queries

- **How can we express queries so that the DBMS can understand and process them automatically?**
  - In natural language? → Still science fiction!
- Relational databases can be queried by different **query languages.**
  - Relational algebra (operational, based on set and relational operators)
  - Relational calculus (declarative, based on logic)
  - SQL: a standard computer language which integrates most of the two previous approaches and *looks like* natural language.

2.2 Relational database goals: Queries

Set operators

- UNION: ∪ : The union of two relations R and S defines a relation that contains all the tuples of R or S or both R and S, duplicate tuples being eliminated.

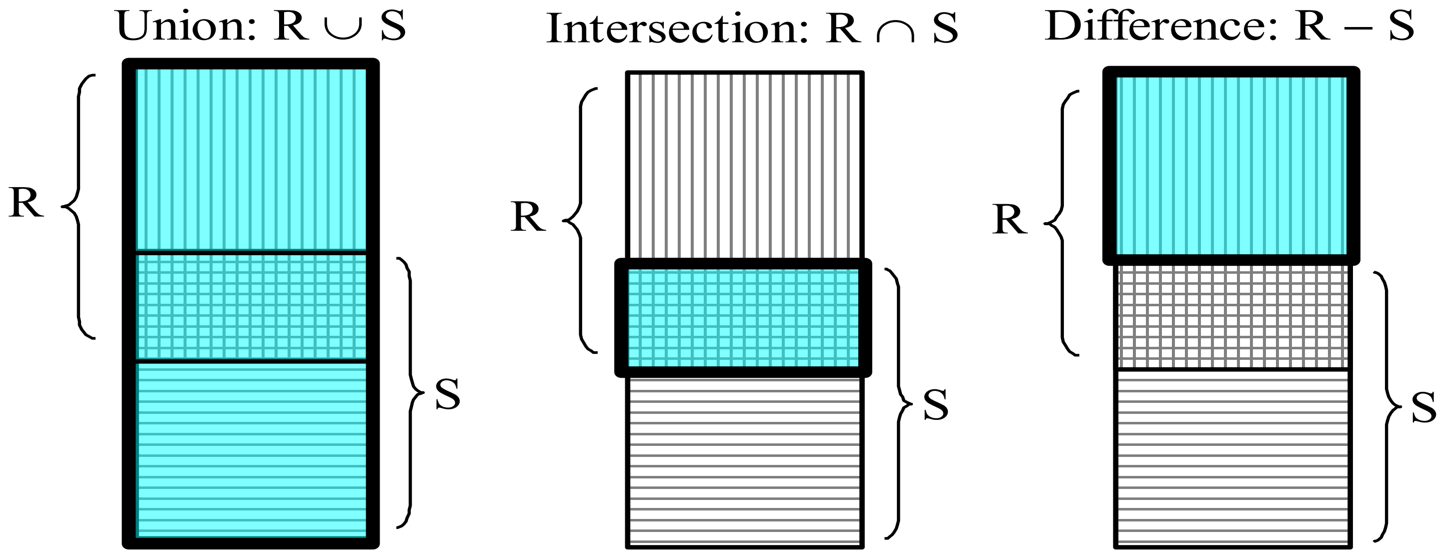
- INTESECTION: ∩ : R ∩ S defines a relation consisting of the set of all tuples that are in both R and S

- DIFFERENCE: – : R – S defines a relation consisting of the tuples that are in relation R, but not in S.

- PRODUCT: × : R x S defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S

2.2 Relational database goals: Queries

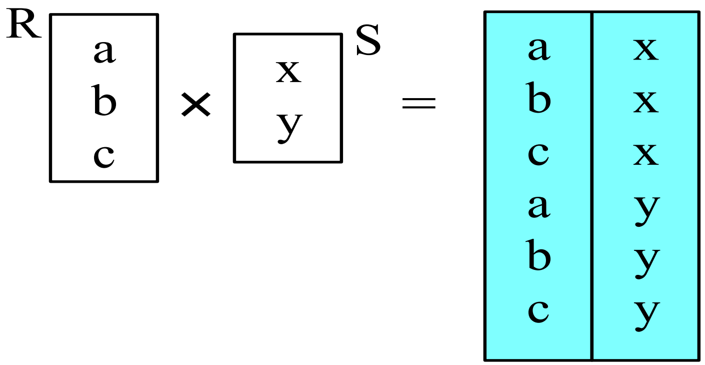
Set operators


Union: R ∪ S

Intersection: R ∩ S

Difference: R – S

Product: R × S



2.2 Relational database goals: Queries

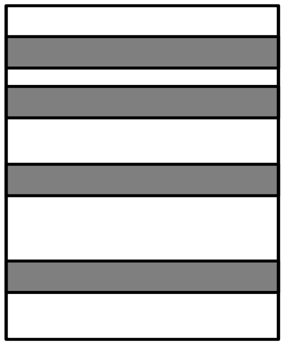
Relational operators.

- SELECTION: WHERE … : selects the tuples that satisfy the specified condition (predicate)
- PROYECTION: […]: extracts the specified attributes (columns) and eliminates duplicates.
- JOIN: ⊗…: defines a relation that contains tuples satisfying some condition from the cartesian product.
- RENAME: (*old*, *new*) : changes the name of a column
- Logical operators, conditions and expressions (AND, OR, NOT),…

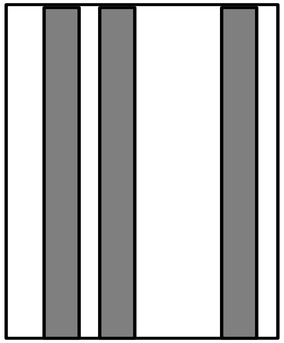
2.2 Relational database goals: Queries

Relational operators.

Selection



Projection



Join

$$\begin{array}{|c|c|} a1 & b1 \\ a2 & b1 \\ a3 & b2 \end{array} \bowtie \begin{array}{|c|c|} b1 & c1 \\ b2 & c2 \\ b3 & c3 \end{array} = \begin{array}{|c|c|c|} a1 & b1 & c1 \\ a2 & b1 & c1 \\ a3 & b2 & c2 \end{array}$$

Exercise 1.1: Relational Algebra

Write expressions to obtain:

1. Name (nombre) of all the subjects.
2. Name (nombre) of the subjects with 4 lab groups (GP)
3. Name (nombre) of lecturers with categoria ‘Titular’ teaching the subject 11545
4. Name (nombre) of the lecturers with categoria “Titular” teaching a subject in the ‘1A’ semester (semestre).
5. Name of lecturers teaching a subject with 2 GT groups

Queries using SQL

SELECT[ALL | DISTINCT] {*expression*$_1$, *expression*$_2$,…*expression*$_n$} | *

FROM table

[WHERE *condition*]

[GROUP BY *condition*] [HAVING *condition*]

[ORDER BY {*column*$_1$, *column*$_2$, … *column*$_m$}]

FROM: Specifies the table/s to be used

WHERE: Filters the rows subject to some condition

GROUP BY: Forms groups of rows with the same column value

HAVING: Filter the groups subject to some condition

SELECT: Specifies which columns are to appear in the output

ORDER BY: Specifies the order of the output

Exercise 1.2: SQL

Write queries in SQL to obtain:

1. Name (*nombre*) of the all the subjects.
2. Name (*nombre*) of the subjects with 4 lab groups (*GP*)
3. Name (*nombre*) of lecturers with *categoria* ‘Titular’ teaching the *subject* 11545
4. Name (*nombre*) of the lecturers with *categoria* “Titular” teaching a subject in the ‘1A’ semester (*semestre*).
5. Name of lecturers teaching a subject with 2 *GT* groups
6. Name (nombre) of lecturers with categoria=‘Titular’ and with no telephone number

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

- 3.1 Data types
- 3.2 Tuple and relation
- 3.3 Null value
- 3.4 Constraints

4 Constraints and transactions

3 The relational data model: Terminology

Common terminology	RDM
table	relation
row / record	tuple
column / field	attribute
data type	domain

But they are not exactly equivalent

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

- 3.1 Data types
- 3.2 Tuple and relation
- 3.3 Null value
- 3.4 Constraints

4 Constraints and transactions

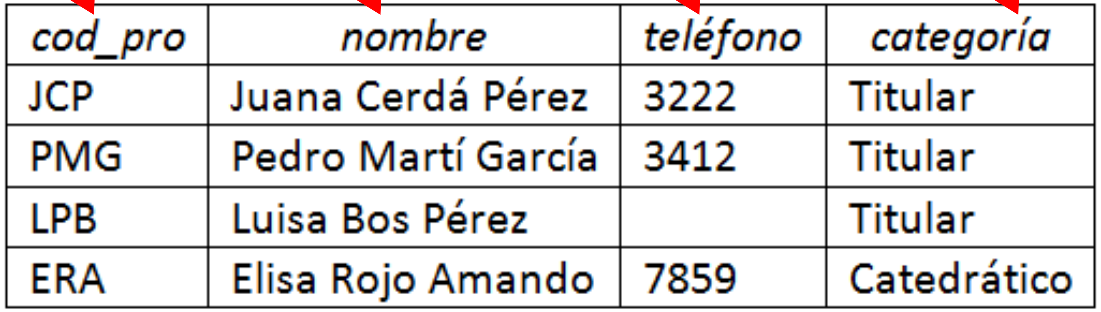
3.1 Data types

- Depend on the Relational DataBase Manager System (DBMS)
- Examples:
  - Numeric: *integer, smallint*, *numeric, number, real, float*.
  - Alfanumeric: chars, *string, varchar*,… i.e. ‘Pepe’.
  - Date
  - …

3.1 Data types

char(3) char(50) char(8) char(15)

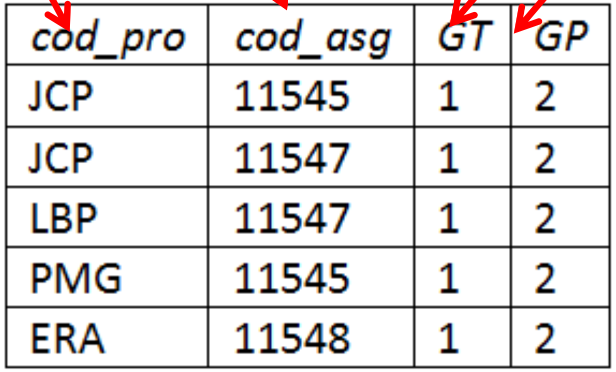
Lecturer



cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

char(3) char(5) smallint

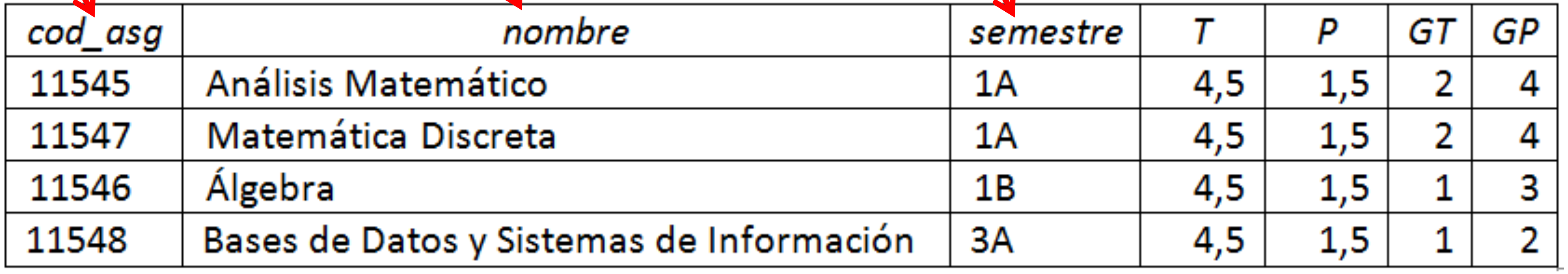
Teaching



cod_pro	cod_asg	GT	GP
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

char(5) char(50) char(2) real smallint

Subject



cod_asg	nombre	semestre	T	P	GT	GP
11545	Análisis Matemático	1A	4,5	1,5	2	4
11547	Matemática Discreta	1A	4,5	1,5	2	4
11546	Álgebra	1B	4,5	1,5	1	3
11548	Bases de Datos y Sistemas de Información	3A	4,5	1,5	1	2

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

- 3.1 Data types
- 3.2 Tuple and relation
- 3.3 Null value
- 3.4 Constraints

4 Constraints and transactions

3.2 Tuple and relation

➡ A tuple schema τ is a set of pairs of the form:

$$\tau = \{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$$

Where:

$\{A_1, A_2, \ldots, A_n\}$ $(n>0)$ is the set of **attribute names** in the schema, necessarily different

$\{D_1, D_2, \ldots, D_n\}$ are the **domains** associated with the above-mentioned attributes.

3.2 Tuple and relation

Example of tuple schema

Person =

{(person_id, integer), (name, char), (address, char)}

where:

{ person_id, name, address } is the set of attribute names in the schema.

integer, char, char are the domains which are associated with the attributes.

3.2 Tuple and relation

tuple ⟷ Row / Record

Tuple:

➡ A tuple t of tuple schema $\tau = \{(A_1, D_1), (A_2, D_2), \ldots, (A_n, D_n)\}$

Is a set of pairs of the form::

$$t = \{(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\}$$

$\forall i\ v_i \in D_i$

3.2 Tuple and relation

Examples of tuples:

Given the following **tuple schema**:

Person = {(person_id, integer), (name, char), (address, char)}

We have some **tuples**:

t_1 = {(person_id, 2544), (name, “Joan Roig”), (address, “Sueca 15”)}

t_2 = {(person_id, “2644F”), (name, “R3PO”), (address, “46022”)} ✗

t_3 = { (name, “Pep Blau”), (person_id, 9525), (address, “dunno!”)}

3.2 Tuple and relation

A **relation** is a set of tuples of the same schema

A **relation schema** is the schema of the tuples composing the relation

Notation:

$R\ (A_1{:}D_1,\ A_2{:}\ D_2,\ldots,\ A_n{:}\ D_n\}$

defines a relation R of schema

$\{\ (A_1,\ D_1),\ (A_2,\ D_2),\ldots,\ (A_n,\ D_n)\ \}$

3.2 Tuple and relation

- Relation schema for the **Teaching** relation:

  {(*cod_pro*, char(3)), (*cod_asg*, char(5)), (*GT*, smallint), (*GP*, smallint)}

- Example of tuple of the **Teaching** relation:

  {(cod_pro, ‘JCP’), (cod_asg, ‘11545’), (GT,1), (GP,2)}

  {(cod_pro, ‘JCP’), (cod_asg, ‘11545’), (GT, ~~‘X’~~), (GP,2)}

- **Teaching relation**:

  {{(cod_pro, ‘JCP’), (cod_asg, ‘11545’), (GT,1), (GP,2)},
  {(cod_pro, ‘JCP’), (GT,1), (cod_asg, ‘11547’), (GP,2)},
  {(GT,1), (cod_pro, ‘LBP’), (cod_asg, ‘11547’), (GP,2)},
  {(cod_pro, ‘PMG’), (cod_asg, ‘11545’), (GT,1), (GP,2)},
  {(cod_asg, ‘11548’), (cod_pro, ‘ERA’), (GT,1), (GP,2)}}

3.2 Tuple and relation

Properties of a relation

- ***Degree*** *of a relation:* Number of attributes of its schema

- ***Cardinality*** *of a relation:* Number of tuples that compose the relation

- ***Compatibility****:* Two relations R y S are compatible if their schemas are identical

3.2 Tuple and relation

Example:

Given the following tuple schema:

Person = {(person_id, integer), (name, char(15)), (address, char(20))}

A relation of the *PERSON* schema might be as follows:

{ { (person_id, 1234), (name, “Pepa Gómez”), (address, “Colón 15”) },
{ (person_id, 2045), (name, “Juan Pérez”), (address, “Cuenca 20”) },
{ (name, “José Abad”), (person_id, 1290), (address, “Blasco Ibáñez 35) },
{ (name, “María Gutiérrez”), (person_id, 35.784.843) (address, “Reina 7”) } }

Degree:

Cardinality:

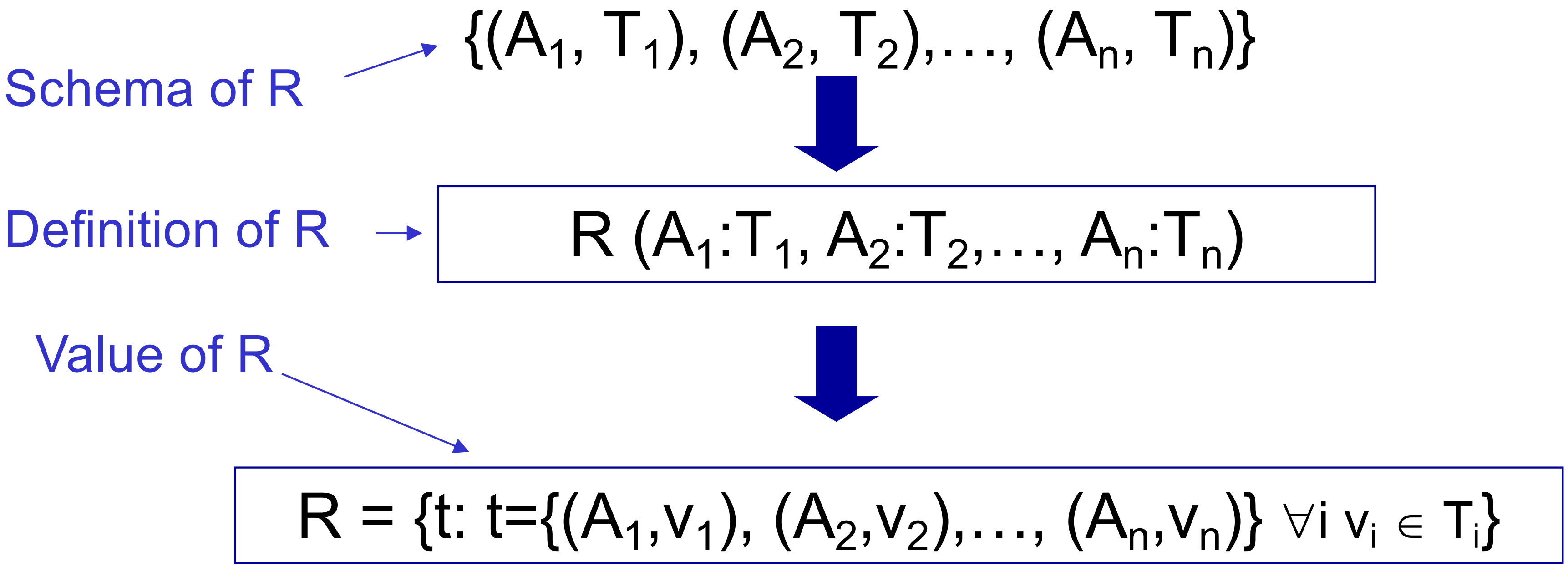
Compatible with:

3.2 Tuple and relation

Relation:

A relation is set of tuples of the same schema, which is called *relation schema*

relation R with a relation schema



Schema of R

$\{(A_1, T_1), (A_2, T_2), \ldots, (A_n, T_n)\}$

Definition of R

$R\ (A_1{:}T_1, A_2{:}T_2, \ldots, A_n{:}T_n)$

Value of R

$R = \{t\text{: } t=\{(A_1,v_1), (A_2,v_2), \ldots, (A_n,v_n)\}\ \forall i\ v_i \in T_i\}$

3.2 Tuple and relation

Representation of a relation → TABLE

- tuples are represented as rows
- attributes give name to the column headers

Example: PERSON relation

Column ≈ Attribute

Person_id	Name	Address
2045	Juan Pérez	Cuenca 20
1290	José Abad	Blasco Ibáñez 35
3578	María Gutiérrez	Reina 7
1234	Pepa Gómez	Colón 15

Row ≈ Tuple

3.2 Tuple and relation

- The Table is only a Matrix Representation of a Relation

- Traits which distinguish a relation (derived from the definition of relation as a set of sets):

  - There can’t be repeated tuples in a relation (a relation is a set).
  - There isn’t a top-down order in the tuples (a relation is a set).
  - There isn’t a left-to-right order in the attributes of a relation (a tuple is a set).

3.2 Tuple and relation

- The set of *relation definitions* which represent an information system is called *relational (logical) schema*.

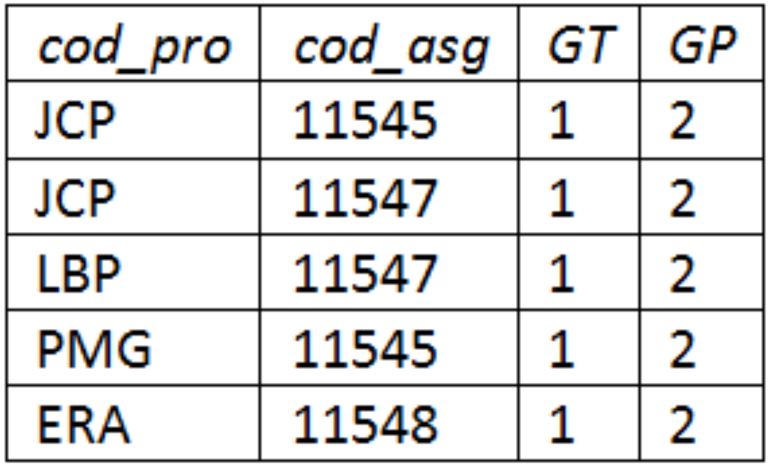
- The content (set of tuples) of the relations of the relational schema is the database

3.2 Tuple and relation

Relation of the schema Teaching:

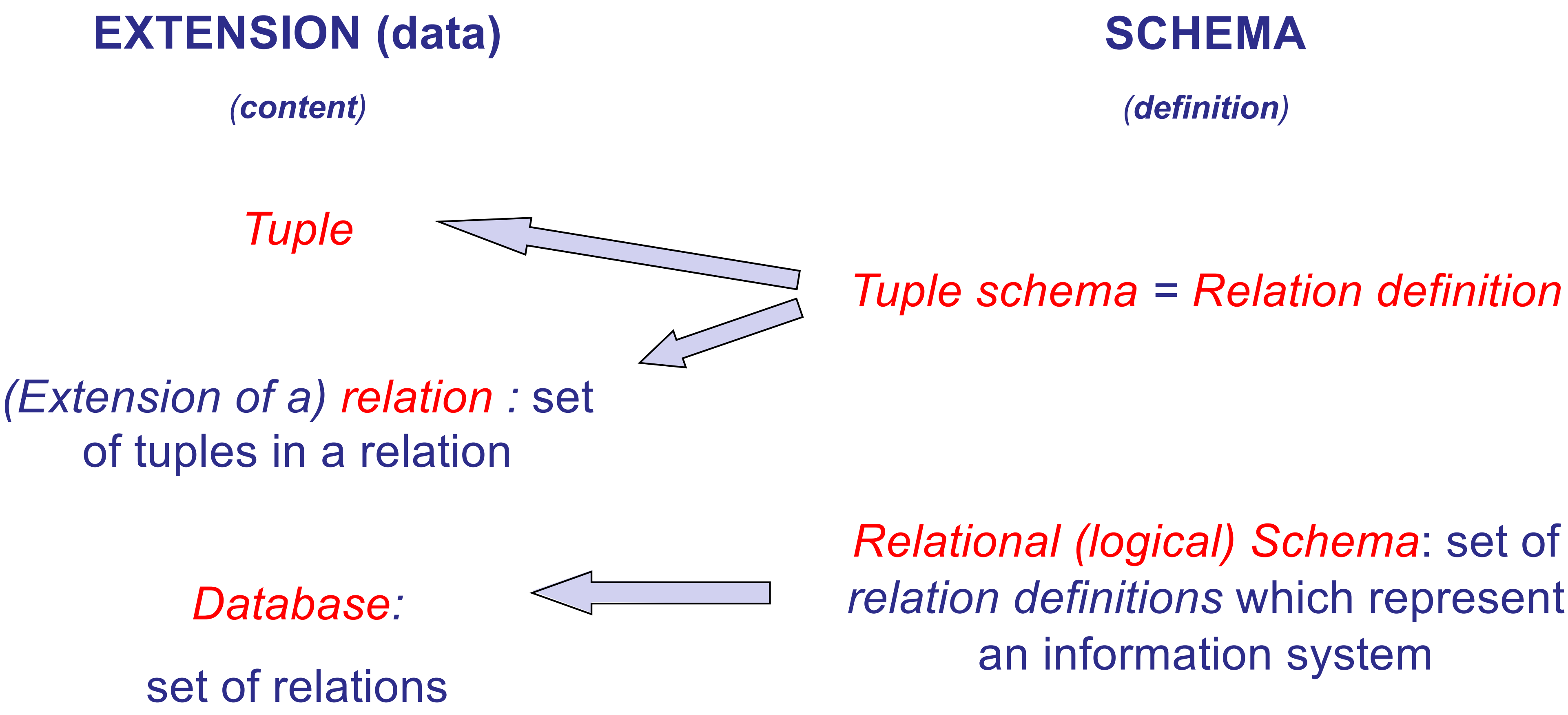
{{(cod_pro, ‘JCP’), (cod_asg, ‘11545’), (GT,1), (GP,2)},
{(cod_pro, ‘JCP’), (GT,1), (cod_asg, ‘11547’),(GP,2)},
{(cod_asg, ‘11547’), (cod_pro, ‘LBP’), (GT,1), (GP,2)},
{(cod_pro, ‘PMG’), (GT,1), (cod_asg, ‘11545’), (GP,2)},
{(cod_pro, ‘ERA’), (cod_asg, ‘11548’), (GT,1), (GP,2)}}

Matrix representation of the relation Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

3.2 Tuple and relation



EXTENSION (data)

*(**content**)*

SCHEMA

*(**definition**)*

Tuple

Tuple schema = *Relation definition*

(Extension of a) relation : set of tuples in a relation

Database:

set of relations

Relational (logical) Schema: set of *relation definitions* which represent an information system

Attention!: DBMSs understand a table as the definition of a relation and not as its content, which eventually changes by applying <u>operators</u>.

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

- 3.1 Data types
- 3.2 Tuple and relation
- 3.3 Null value
- 3.4 Constraints

4 Constraints and transactions

3.3 Null value

What happens if we don’t know the value a tuple takes in some of its attributes?

- Solution in Programming Languages:

  use of special or extreme values (-1, “Empty”, “ ”, “We don’t know”, 0, “No address”, “---”, ...)

It is only a representation

- Solution in the Relational Model: **NULL VALUE (?)**

  A Domain is something more than a datatype: A domain is a set of elements which always includes the NULL value.

3.3 Null value

Given the domains:

id_dom: integer
name_dom, add_dom: char(20)

Tuple schema:

Person = {(person_id, id_dom), (name, name_dom), (address, add_dom)}

Tuples:

t_1 = { (person_id, 12345678), (name, “Pepa Gómez”), (address, “Paz 10”) }
t_2 = { (name, “Pep Blau”), (person_id, 9525869), (address, **?**) }

We say that t_2.address **is null**, *not that* ~~t_2.address = null~~.

- We may use the operator “**isnull**” to check:

isnull (t_2.address)

3.3 Null value

The null value represents that there is no known value, so

If t_2.address is NULL,

What is the result of t_2.address= “Sesamo Street” ?

→ It’s neither true nor false because it is undefined

We need a tri-valued logic:

- True
- False
- Undefined

3.3 Null value

Example:

t = {(cod_pro, ‘LBP’), (nombre, ‘Luisa Bos Pérez’), (telefono, ?), (categoria, ‘Titular’)}

- t.cod_pro= ‘LBP’ = true
- t.categoría <> ‘Titular’ = false
- t.telefono = ‘55544’ = ***undefined***

3.3 Null value

This applies to all the comparison operators $<, >, =, \geq, \leq, \neq$

Evaluation:

A α B (where α is a comparison operator) could be evaluated as undefined if at least one A or B is **null**; otherwise it is evaluated to the certainty value of the comparison A α B

Examples:

2 < 5 → true

3 < ? → undefined

? < ? → undefined

IsNull(3) → false

IsNull(?) → true

3.3 Null value: AND, OR, NOT

G	H	G $\wedge$ H	G $\vee$ H
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true
undefined	undefined	undefined	undefined
undefined	false	false	undefined
undefined	true	undefined	true
false	undefined	false	undefined
true	undefined	undefined	true

G	$\neg$G
false	true
undefined	undefined
true	false

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

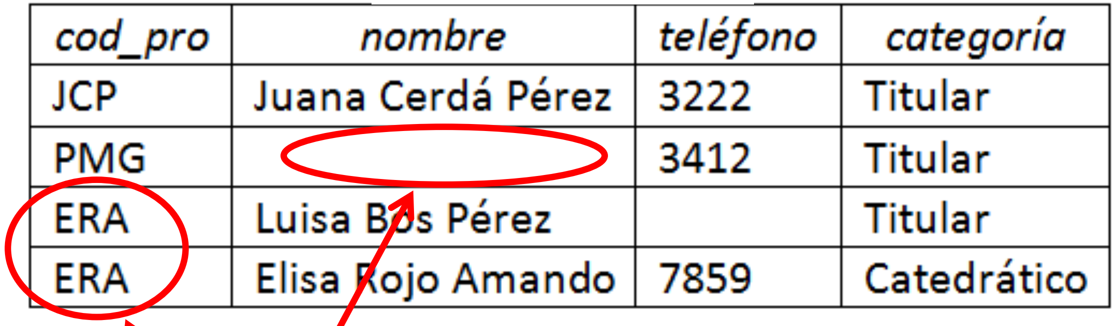
- 3.1 Data types
- 3.2 Tuple and relation
- 3.3 Null value
- 3.4 Constraints

4 Constraints and transactions

3.4 Constraints

Is this a valid representation of reality?

Lecturer

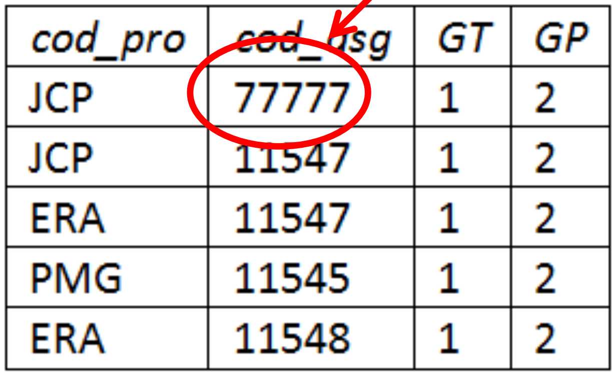


cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG		3412	Titular
ERA	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

? ?

?

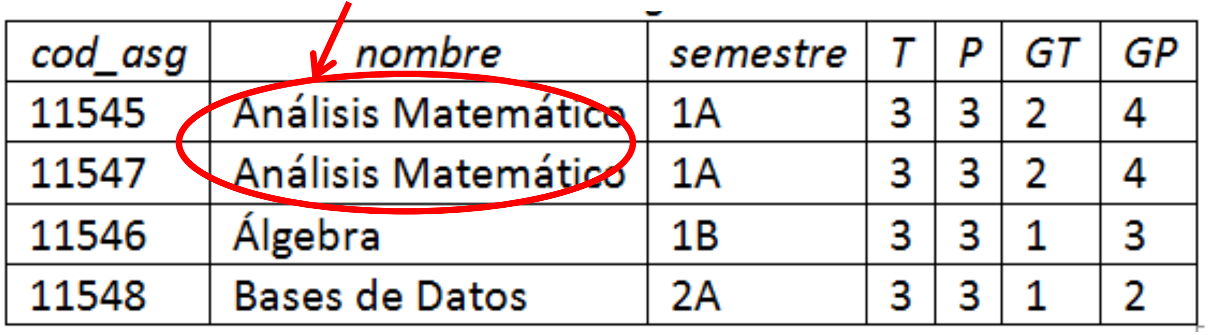
Teaching



cod_pro	cod_asg	GT	GP
JCP	77777	1	2
JCP	11547	1	2
ERA	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

?

Subject



cod_asg	nombre	semestre	T	P	GT	GP
11545	Análisis Matemático	1A	3	3	2	4
11547	Análisis Matemático	1A	3	3	2	4
11546	Álgebra	1B	3	3	1	3
11548	Bases de Datos	2A	3	3	1	2

No, this is a non-valid representation of reality

3.4 Constraints

Solution

- Definition of *domains*
- *Uniqueness constraints.*
- *Not null constraints.*
- Definition of *primary keys.*
- Definition of *foreign keys*.
- General integrity constraints.

They are specified together with the **database schema**.

The responsible for ensuring them is the **DBMS**.

3.4 Constraints

✓*cod_pro* identifies lecturers

→ Primary key

✓*nombre* is unique for each subject

→ Uniqueness

✓The name (*nombre*) of a lecturer must be known

→ Not null value

✓*cod_asg* in *Teaching* refers to an existing subject

→ Foreign key (referential integrity)

3.4 Constraints

Not null

NNV: $\{A_1, \ldots, A_p\}$

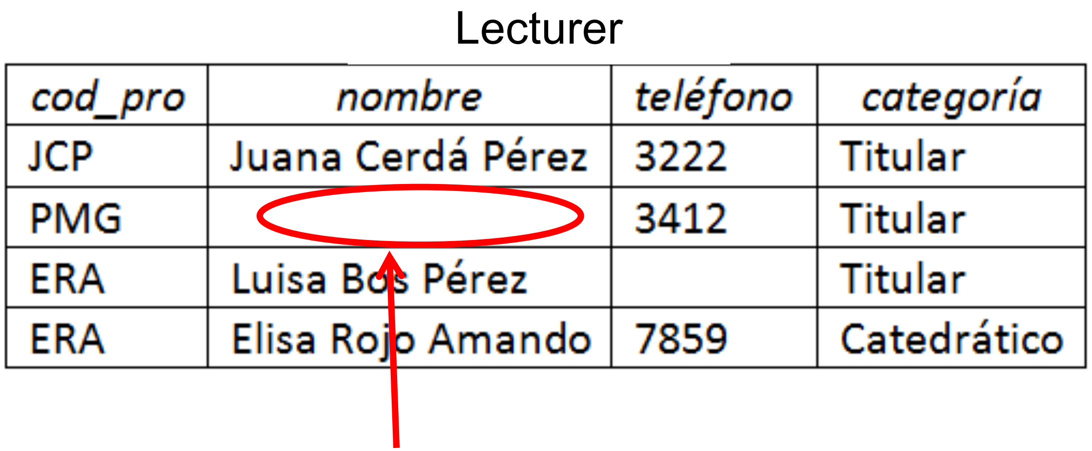
The definition of a not null constraint over a set of attributes $K=\{A_1, A_2, \ldots A_3\}$ of a relation R expresses the following property:

"There cannot be a tuple in R having the null value in any attribute of K"

$$\forall\, t\, (\, t \in R \rightarrow\ \neg\exists\, Ai \in K\ \wedge\ Null(t.Ai))$$

3.4 Constraints

Not null



Lecturer

cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG		3412	Titular
ERA	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Not allowed

Lecturer (cod_pro: char(3), nombre: char(50),
telefono: char(8), categoria:char(15))

NNV:{nombre}

"there cannot be a tuple in Lecturer *which has the null value in the* nombre *attribute".*

3.4 Constraints

Uniqueness

UNI: $\{A_1, \ldots, A_p\}$

The definition of a uniqueness constraint over a set of attributes $K=\{A_1, A_2, \ldots A_3\}$ of a relation R expresses the following property:

"There cannot be two tuples in R having the same value in all the attributes of K"

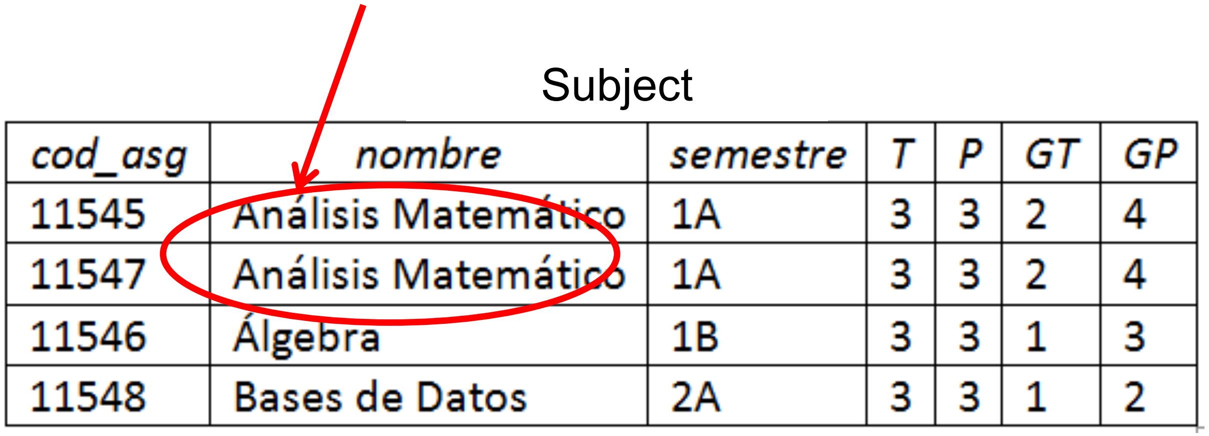
$$\neg\exists t_1\ \exists t_2\ (t_1 \in R \land t_2 \in R \land t_1 \neq t_2 \land \forall A_i\ (A_i \in K \rightarrow t_1.A_i = t_2.A_i))$$

3.4 Constraints

Uniqueness

Not allowed

Subject



cod_asg	*nombre*	*semestre*	*T*	*P*	*GT*	*GP*
11545	Análisis Matemático	1A	3	3	2	4
11547	Análisis Matemático	1A	3	3	2	4
11546	Álgebra	1B	3	3	1	3
11548	Bases de Datos	2A	3	3	1	2

Subject (cod_asg: char(5), nombre: char(50),
semestre: char(2), T: real, P: real,
GT: smallint, GP: smallint)

UNI:{nombre}

"There cannot be two tuples in Subject *which have the same value for the attribute* nombre*".*

3.4 Constraints

Primary key

PK: $\{A_1,\ldots, A_p\}$

Given a set of attributes $K=\{A_1, A_2,\ldots A_3\}$ which has been defined as primary key for R, we say that R satisfies the primary key constraint if the following properties hold:

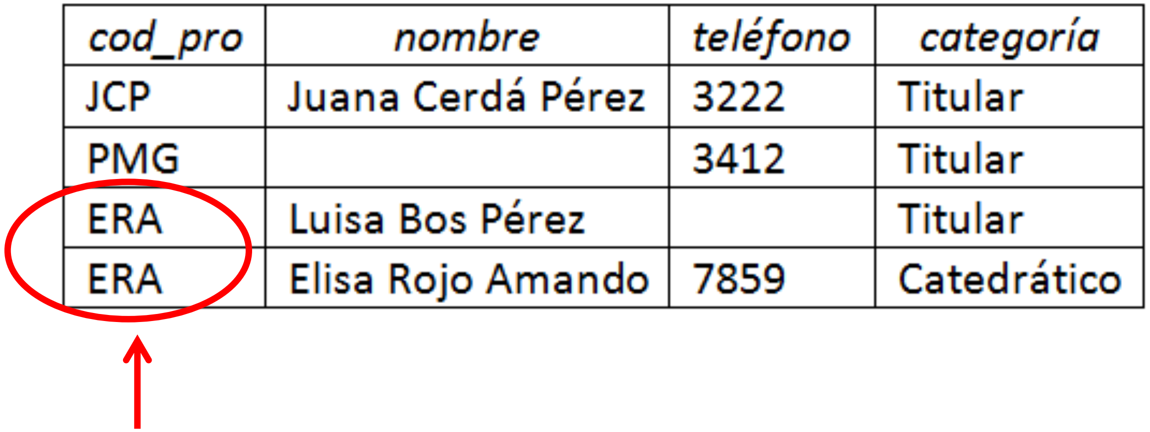
1. **R satisfies a not null constraint over PK**
2. **R satisfies the uniqueness constraints over PK**

Note that PK must be minimal: There cannot be any proper subset that could also be primary key for R

3.4 Constraints

Primary key

Lecturer



cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG		3412	Titular
ERA	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Not allowed

Lecturer (cod_pro: char(3), nombre: char(50),
telefono: char(8), categoria:char(15))

PK:{cod_pro}

"cod_pro *is the primary key for* Lecturer"

3.4 Constraints

Foreign key

- The use of **foreign keys** is the mechanism provided by the relational model to express associations between the objects in a database schema. This mechanism is defined such that these associations, if performed, would be carried out adequately.

  - With this goal, we can add to the schema of a relation S, a set of attributes which refer to a set of attributes of a relation *R*.
  - This set of attributes K = {A_1,…, A_p} is called ***foreign key** in relation S which refers to relation R*.

3.4 Constraints

Foreign key

$$\boxed{FK: \{A_1,\ldots, A_p\} \rightarrow R}$$

Given a foreign key FK **in S which refers to R**, this is defined as:

1. A set of attributes $K=\{A_1, A_2,\ldots A_3\}$ in the schema of S
2. A bijection f: $K\rightarrow J$ *such as:*
   - J is a set of attributes in R
   - J has a **uniqueness** constraint
   - $\forall A_i \in K \rightarrow$ A_i and $f(A_i)$ have the **same domain**
3. A type of referential integrity:
   - weak
   - partial
   - full / complete

3.4 Constraints

FK: $\{A_1\} \rightarrow R$

If FK = $\{A_1\}$ (contains **only one attribute**) the three types of referential integrity match:

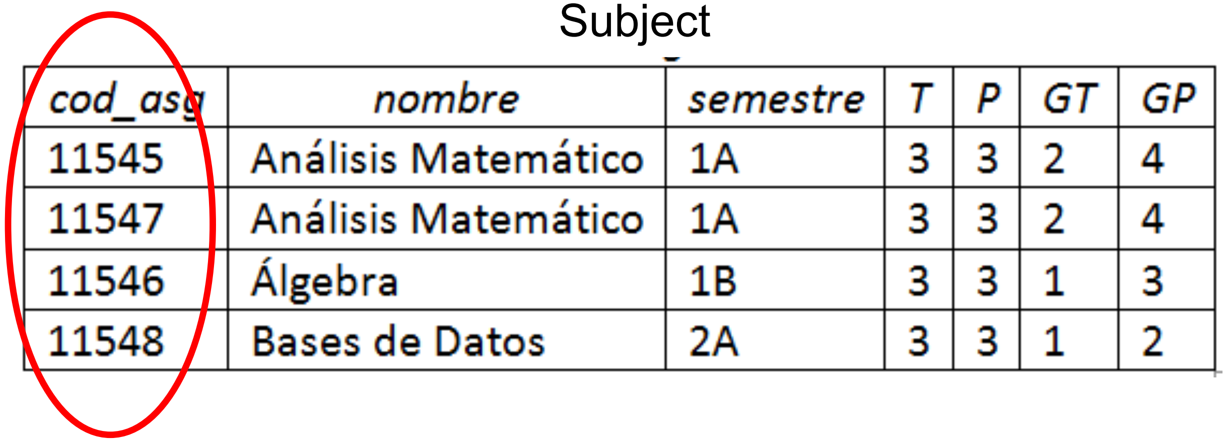
S satisfies the referential integrity constraint if all tuple in S met:

- A_1 is NULL, or
- There is one tuple (and only one) in R with the same value in the $f(A_1)$ attribute than A_1 in S

3.4 Constraints

Foreign key

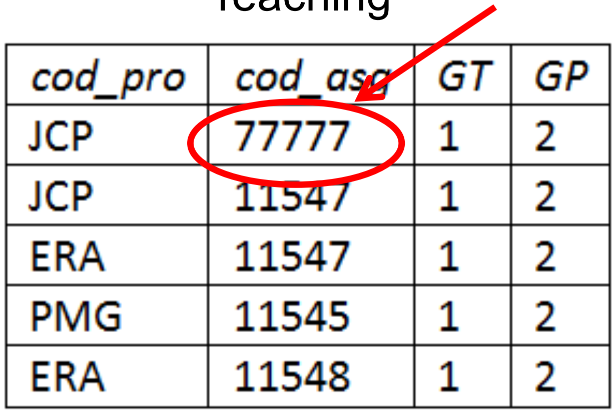
Subject



cod_asg	nombre	semestre	T	P	GT	GP
11545	Análisis Matemático	1A	3	3	2	4
11547	Análisis Matemático	1A	3	3	2	4
11546	Álgebra	1B	3	3	1	3
11548	Bases de Datos	2A	3	3	1	2

Teaching

?



cod_pro	cod_asg	GT	GP
JCP	77777	1	2
JCP	11547	1	2
ERA	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

Subject (cod_asg: char(5), nombre: char(50), semestre: char(2),
T: real, P: real, GT: smallint, GP: smallint)

Teaching (cod_pro: char(3), cod_asg: char(5), GT: smallint, GP: smallint)
FK:{cod_asg} → Subject

“If there is a tuple in Teaching *such that the value* cod_asg *is not null, then there must be one (and only) one tuple in* Subject *such that the value of* cod_asg *in* Teaching *matches the value* cod_asg *in* Subject*”*

3.4 Constraints

FK: $\{A_1,\ldots,A_p\} \rightarrow R$

If K has more than one attribute S satisfies the referential integrity if the following property is met:

Weak R.I.:

"If in a tuple of *S* all the values for the attributes in *K* have a non-null value, then there must exist a tuple in *R* taking the same values for the attributes in *J* as the attributes in *K*"

$$\forall t\,(\, t \in S \rightarrow (\,\exists A_i\,(A_i \in K \wedge \mathrm{IsNull}(t.A_i)) \vee \exists m\,(m \in R \wedge \forall A_i\,(\,A_i \in K \rightarrow t.A_i = m.f(A_i)\,))))$$

3.4 Constraints

Foreign key

$\text{FK: } \{A_1,\ldots, A_p\} \rightarrow R$

If K has more than one attribute S satisfies the referential integrity if the following property is met:

Partial R.I.:

"If in a tuple of *S* one or more attributes in *K* have a non-null value, then there must exist a tuple in *R* taking the same values for the attributes in *J* as the values in the non-null attributes in *K*."

$$\forall\, t\, (\, t \in S \rightarrow (\, \forall A_i\, (A_i \in K \rightarrow \text{IsNull}\,(t(A_i)))$$

$$\vee$$

$$\exists\, m\, (m \in R \wedge \forall A_i\, ((A_i \in K \wedge \neg \text{IsNull}(t(A_i))\,) \rightarrow t(A_i) = m(f(A_i)))\,))))$$

3.4 Constraints

$$\text{FK: } \{A_1, \ldots, A_p\} \rightarrow R$$

If K has more than one attribute S satisfies the referential integrity if the following property is met:

Complete (or Full) R.I.:

"In any tuple of *S* all the attributes in K have a null value, or none of the attributes in K has a null value. In the latter case, there must exist a tuple in *R* taking the same values for the attributes in *J* as the attributes in *K*."

$$\forall\, t\, (\, t \in S \rightarrow (\, \forall A_i\, (A_i \in K \rightarrow \mathrm{IsNull}(t(A_i)))$$

$$\vee$$

$$\exists\, m\, (m \in R \wedge \forall A_i\, (A_i \in K \rightarrow (\neg \mathrm{IsNull}(t(A_i)) \wedge t.A_i = m(f(A_i)))))))$$

Foreign key: Simplified Notation

- The bijection *f*: *K*→ *J* can be omitted when *J* is **the** primary key **of *R*** and we have one of the following two cases:
  - The set ***K*** has **only one attribute**, or
  - the bijection is defined by the lexical **equality** between the attribute **names** in *K* and *J*.
- The type of referential integrity (weak, partial, complete) can be omitted in any of these cases:
  - The foreign key ***K* has only one attribute**, or
  - **All** the attributes in K have a **not null** constraint,

Since in these cases the three types of referential integrity match.

3.4 Constraints

Foreign key

Office (*code*: dom1, *building*: dom2, *capacity:dom3*)
PK: {code, building}

Telephone(*number*: dom4, *code*: dom1, *building*: dom2)
PK: {number}
FK: {code, building} -> Office

NNV:{code,building}

If *code* and *building* in Telephone *has the* NNV :

NNV (code, building)

the three types of referential integrity are equivalent

3.4 Constraints

Foreign key

Office (*code*: dom1, *building*: dom2, *capacity:dom3*)
PK: {code, building}

Telephone(*number*: dom4, *code*: dom1, *building*: dom2)
PK: {number}
FK: {code, building} -> Office

Weak R.I.

If *there is a tuple in the* Telephone *relation with some (at least one) of the two attributes (*code *or* building*) with a NULL value, the DBMS will not check anything in that tuple.*

3.4 Constraints

Foreign key

Office (*code*: dom1, *building*: dom2, *capacity:dom3*)
PK: {code, building}

Telephone(*number*: dom4, *code*: dom1, *building*: dom2)
PK: {number}
FK: {code, building} -> Office

Partial R.I.

If *there is a tuple in the* Telephone *relation with some (at least one) of the two attributes (*code *or* building*) with a NULL value, the DBMS will only check that there is one tuple in the* Office *relation with the same value for the attributes that are not null in that tuple of* Telephone*.*

3.4 Constraints

Foreign key

Office (*code*: dom1, *building*: dom2, *capacity:dom3*)
PK: {code, building}

Telephone(*number*: dom4, *code*: dom1, *building*: dom2)
PK: {number}
FK: {code, building} -> Office

Full R.I.

If *there is a tuple in the* Telephone *relation with some of the two (not both) attributes* (code *or* building) *with a NULL value, the DBMS will detect a violation of the integrity.*
The referential integrity will not be violated if both attributes (code *and* building) *of* Telephone *have the NULL value in that tuple.*

Violation of the referential integrity

Given two relations R y S such that S has a foreign key K which refers to the attributes J in R, the only operations which may violate their referential integrity are:

- **Operations over *S:***
  - –*Insert a tuple in S*
  - –*Modify some attribute in K in a tuple of S*
- **Operations over *R:***
  - –*Delete a tuple in R*
  - –*Modify* some attribute in *J* in a tuple of *R*

3.4 Constraints

Foreign key

If any of those operations attempts to break the referential integrity, the DBMS aborts the operations (by-default behavior)

But there are other options that can be applied by the DBMS if the foreign key has been previously defined in that way:

- Setting values to **null**

or

- Applying the operation in **cascade**

3.4 Constraints

The referential integrity defined by a foreign key is always preserved but can be done in different ways depending on the foreign key definition:

- Reject the operation (default option).
- Perform the operation but set some values to null to restore integrity.
- Perform the operation but propagate the action in cascade to restore integrity

Options to ensure referential integrity

DELETE:

- **Restrictive** deletion (default option in SQL)
- On delete **cascade**
- On delete **set** to **nulls**

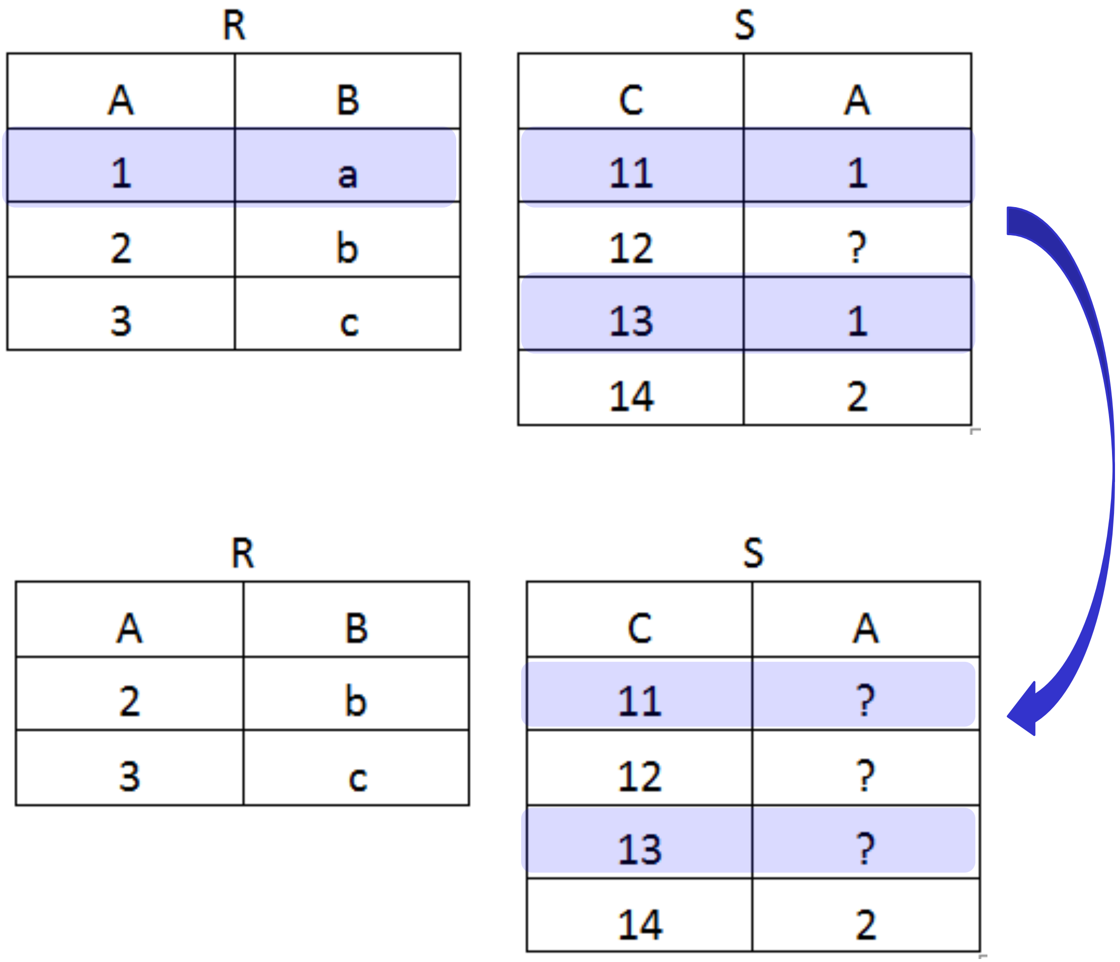
UPDATE:

- **Restrictive** update (default option in SQL)
- On update **cascade**
- On update **set** to **nulls**

3.4 Constraints

Set to nulls

RI:{A} →R



R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Delete tuples from R where A=1

R

A	B
2	b
3	c

S

C	A
11	?
12	?
13	?
14	2

3.4 Constraints

On delete cascade RI:{A} →R

R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Delete tuples from R where A=1

R

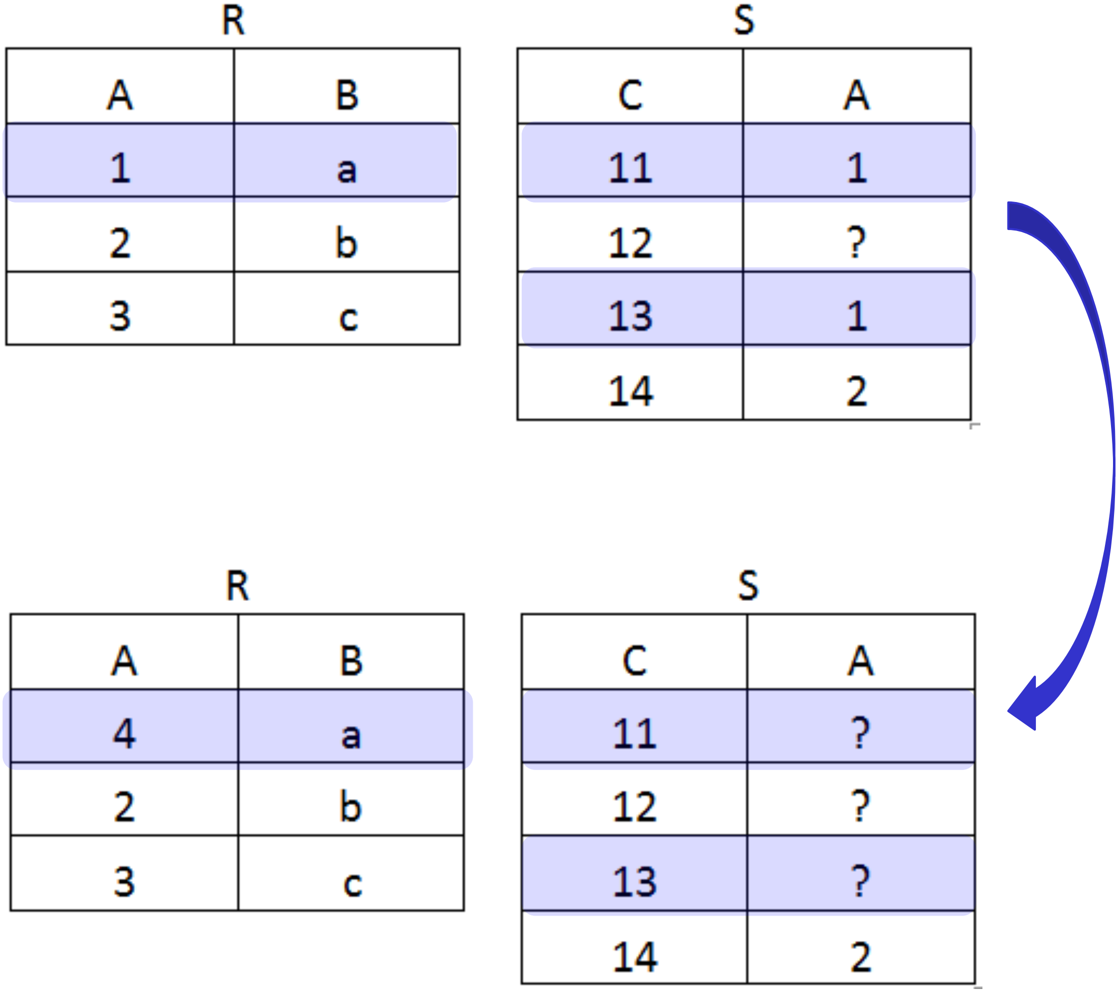
A	B
2	b
3	c

S

C	A
12	?
14	2

3.4 Constraints

On update set to nulls   RI:{A} →R



R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Update tuples in R where A=1
set A = 4

R

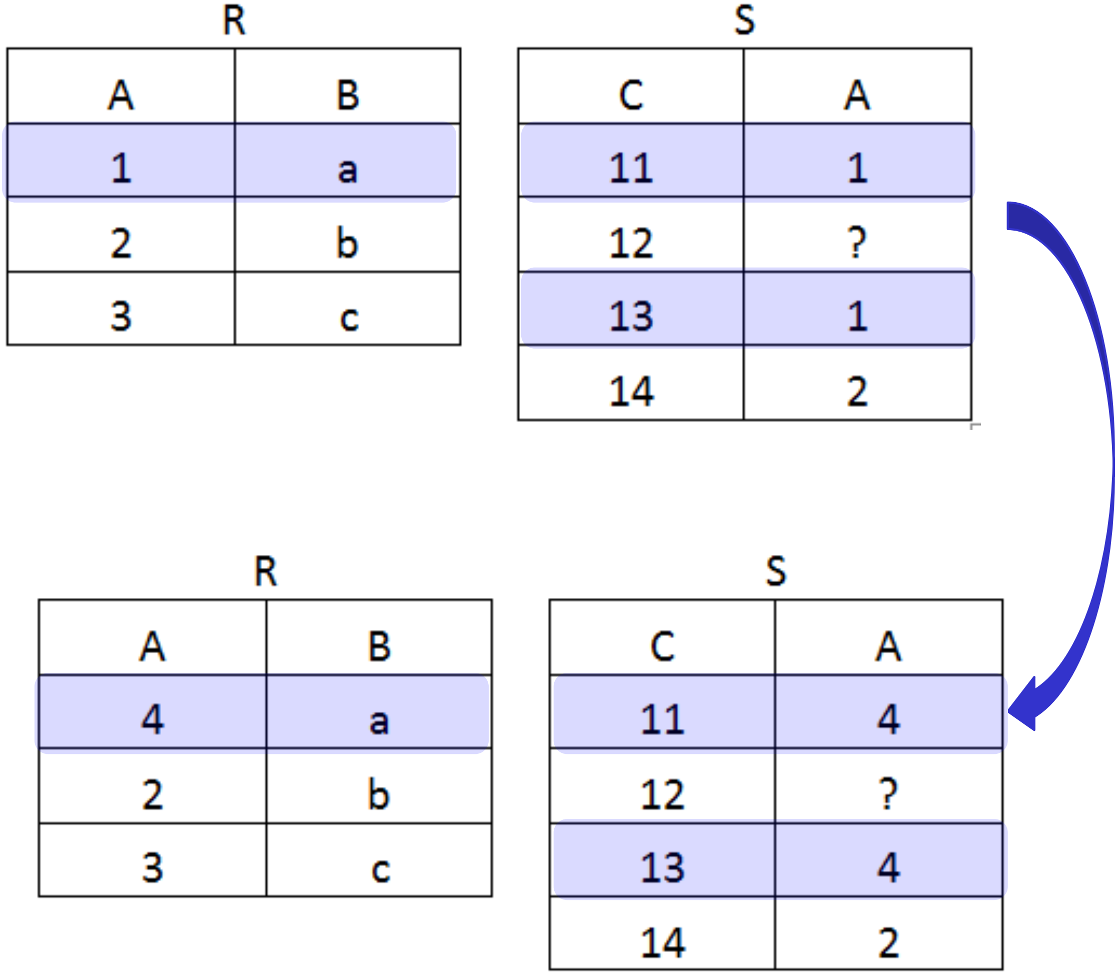
A	B
4	a
2	b
3	c

S

C	A
11	?
12	?
13	?
14	2

3.4 Constraints

On update cascade RI:{A} →R



R

A	B
1	a
2	b
3	c

S

C	A
11	1
12	?
13	1
14	2

Update tuples in R where A=1
set A = 4

R

A	B
4	a
2	b
3	c

S

C	A
11	4
12	?
13	4
14	2

3.4 Constraints

General integrity constraints:

Are those constraints which cannot be expressed by the predefined constraints seen before. They can be:

- Static integrity constraints:

  Affecting one table: attribute or table constraints (usually represented with “CHECK”)

  Affecting several tables: can be expressed with “CREATE ASSERTION ...” or with triggers.

- Transition integrity constraints: triggers.

A database is **valid (it is in a consistent state)**, if all the defined integrity constraints are satisfied.

The DBMS ensures that every update in the database generates a new extension which satisfies all the constraints.

3.4 Constraints

Examples:

- One attribute constraint:

  The value of a semester must be in ('1A', '2A', '3A', '4A', '1B', '2B', '3B','4B')

- Constraints over more than one attribute of the same relation:

  One subject can not have more lab credits (P) than theory credits (T).

- General constraints:

  Affecting more than one table. Sometimes expressed in natural language (English,…).

  All lecturer must teach at least one subject.

Example of relational (logical) schema

Lecturer (**cod_pro:** char(3), **nombre**: char(50), **teléfono**: char(8),
categoría:char(15))

PK:{cod_pro} NNV:{nombre}

Subject (**cod_asg**: char(5), **nombre**: char(50), **semestre**: char(2),
T: real, **P**: real, **GT**: smallint, **GP**: smallint)

PK:{cod_asg} NNV:{nombre,semester,T,P}

UNI:{nombre} RI_1 : (T<=P)

RI_2 : (semestre ∈ {‘1A’,‘1B’,‘2A’,‘2B’,‘3A’,‘3B’,‘4A’,‘4B’})

Teaching (**cod_pro**: char(3), **cod_asg**: char(5), **GT**: smallint,
GP: smallint)

PK:{cod_pro,cod_asg}

FK:{cod_pro} → Lecturer

On delete cascade. On update cascade

FK:{cod_asg} → Subject

Restrictive deletion On update cascade

GC_1: “All lecturer must teach at least one subject”.

Unit 1.2 The Relational Data Model

1 Introduction

2 Introduction to relational databases

3 The relational data model

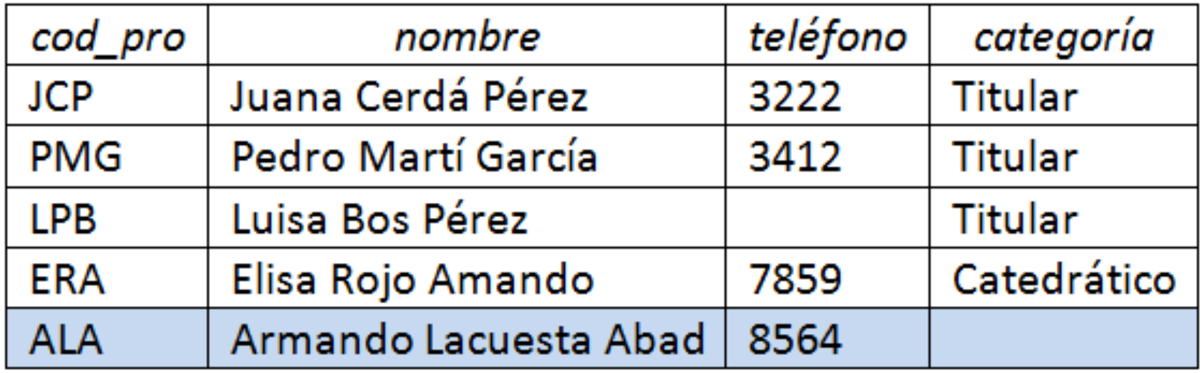
4 Constraints and transactions

4 Constraints and transactions

Add to the Database:

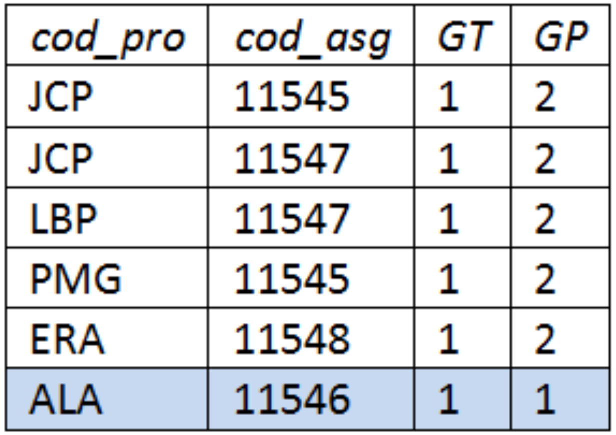
“There is a new lecturer who has the code ‘ALA’, named ‘Armando Lacuesta Abad’, with phone 8564, and with an unknown categoria. He will teach 1 GT and 1 GP of the subject ‘11546’ ”.

Lecturer



cod_pro	*nombre*	*teléfono*	*categoría*
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático
ALA	Armando Lacuesta Abad	8564	

Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2
ALA	11546	1	1

GC$_1$: “All lecturer must teach at least one subject”.

4 Constraints and transactions

Constraints must always be satisfied.

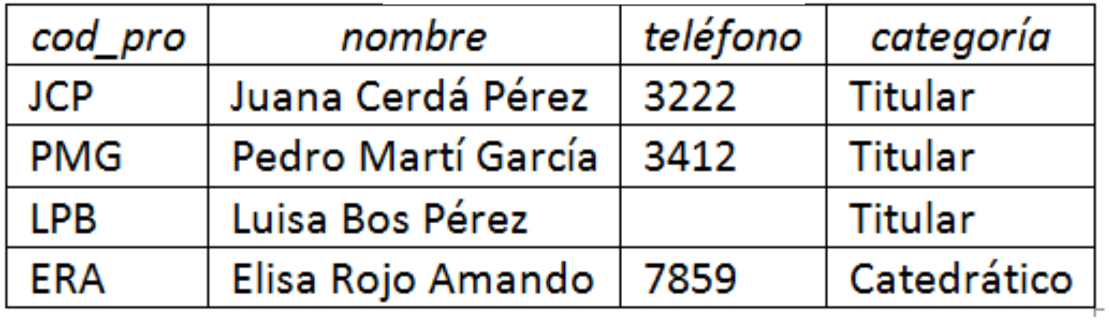
How can we insert the new lecturer ?

A. If we insert the lecturer and then insert his teaching assignments, GC_1 will be violated.

B. If we insert his teaching assignment and then the lecturer, the FK will be violated.

4 Constraints and transactions

Lecturer

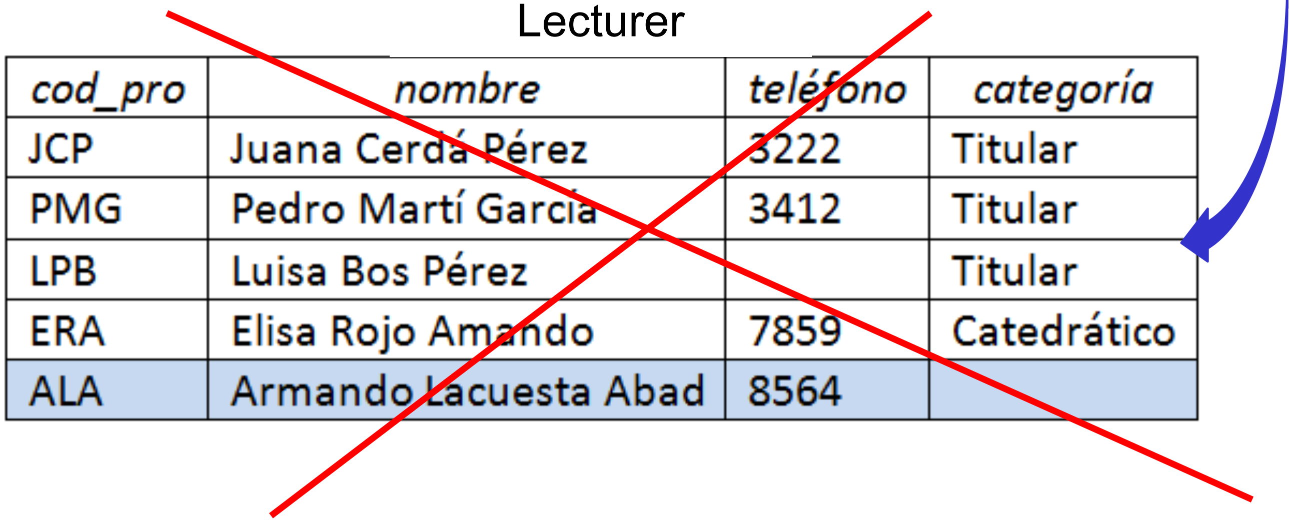


cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

Insert the row:

cod_pro=‘ALA’
nombre=‘Armando …’
teléfono: 8564

Lecturer



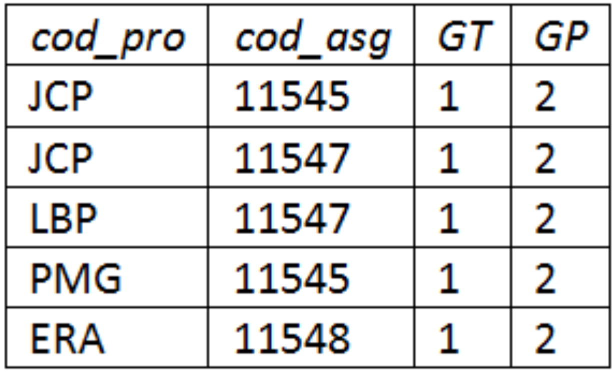
cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático
ALA	Armando Lacuesta Abad	8564	

GC$_1$ is violated. The DBMS rejects the insertion.

GC$_1$: “All lecturer must teach at least one subject”.

4 Constraints and transactions

Teaching

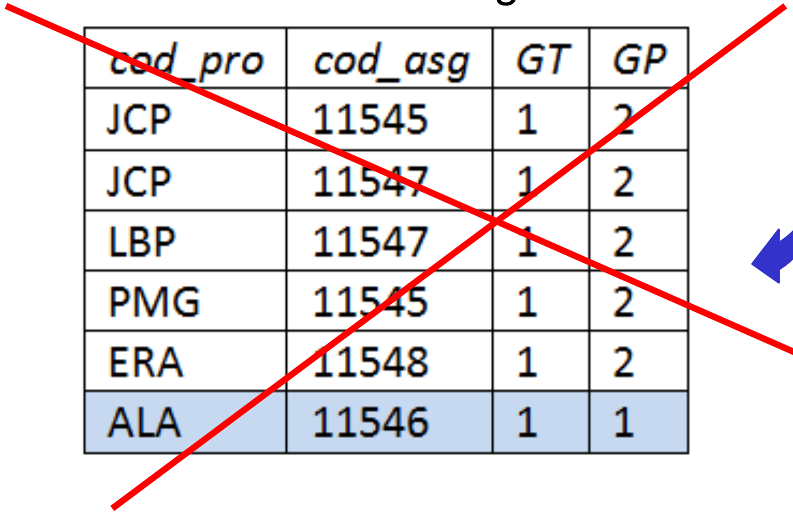


cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

Insert the row:

cod_pro='ALA'
cod_asg: 11546
GT: 1
GP: 1

Teaching



cod_pro	*cod_asg*	*GT*	*GP*
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2
ALA	11546	1	1

The referential integrity is violated.
The DBMS rejects the insertion.

FK:{cod_pro} → Lecturer (cod_pro}

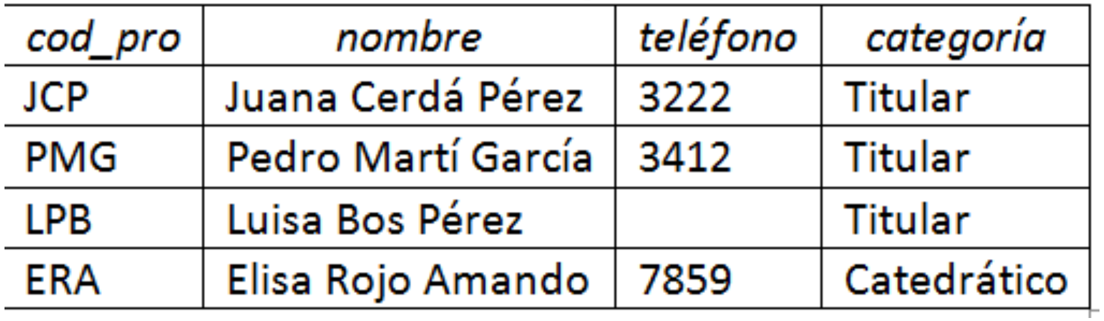
4 Constraints and transactions

A transaction is a sequence of [manipulation or query] operations which constitutes a **logical execution unit**

- We can put a batch of single operations into a transaction (by using appropriate commands).
- Constraints can be disabled during a transaction:
  - Some constraints are evaluated after every single atomic operation (immediate evaluation).
  - Some constraints are evaluated after the transaction is completed (deferred evaluation).
- The database designer or manager are responsible for determining the mode (*immediate* or *deferred*) of each constraint in the system.

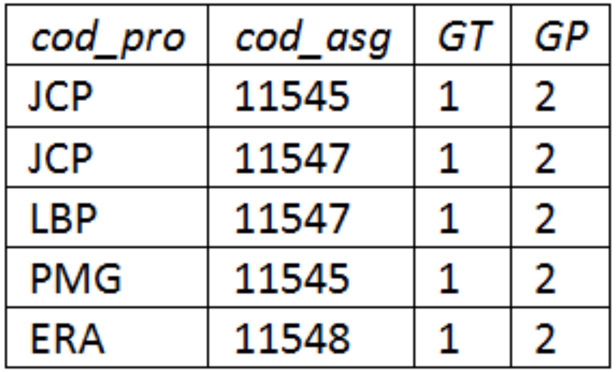
4 Constraints and transactions

Lecturer



cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático

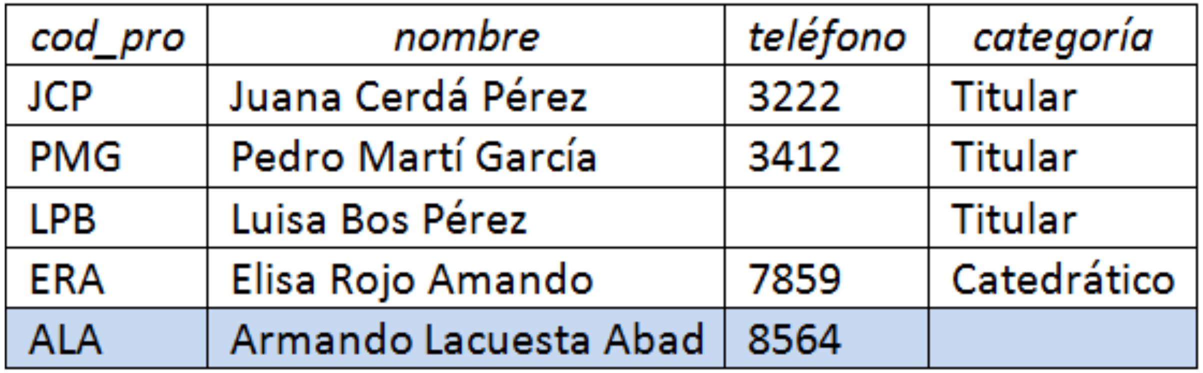
Teaching



cod_pro	cod_asg	GT	GP
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2

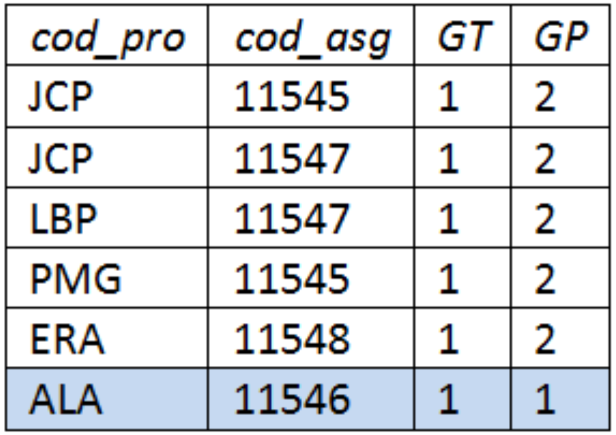
```
INIT TRANSACCTION
   INSERT INTO Lecturer
         {(cod_pro, ‘ALA’), (nombre, ‘Armando Lacuesta Abad’),(teléfono, 8564), (categoría, ?)};
   INSERT INTO Teaching
         {(cod_pro, ‘ALA’), (cod_asg, ‘11546’), (GT, 1), (GP, 1)}
END TRANSACCTION
```

Lecturer



cod_pro	nombre	teléfono	categoría
JCP	Juana Cerdá Pérez	3222	Titular
PMG	Pedro Martí García	3412	Titular
LPB	Luisa Bos Pérez		Titular
ERA	Elisa Rojo Amando	7859	Catedrático
ALA	Armando Lacuesta Abad	8564	

Teaching



cod_pro	cod_asg	GT	GP
JCP	11545	1	2
JCP	11547	1	2
LBP	11547	1	2
PMG	11545	1	2
ERA	11548	1	2
ALA	11546	1	1

Exercise 1.3

Office (*code*: dom1, *building*: dom2, *capacity*: dom3)
PK: {code, building}

Telephone (*number*: dom4, *code*: dom1, *building*: dom2)
PK: {number}
FK: {code, building} -> Office **Weak RI**

On delete set nulls
On update cascade

Office

code	building	capacity
228	1F	1
010	1F	5
228	1G	1
234	2G	2

Telephone

number	code	building
3541	228	1F
3540	010	1F
3202	228	1G

1.- DELETE FROM *Office* WHERE *capacity > 2*
2.- UPDATE *Office* SET *building = 1G* WHERE *capacity >=5*
3.- UPDATE *Office* SET *building = 1G* WHERE *building=1F*
4.- DELETE FROM *Telephone* WHERE number = *3541*