Relational Databases

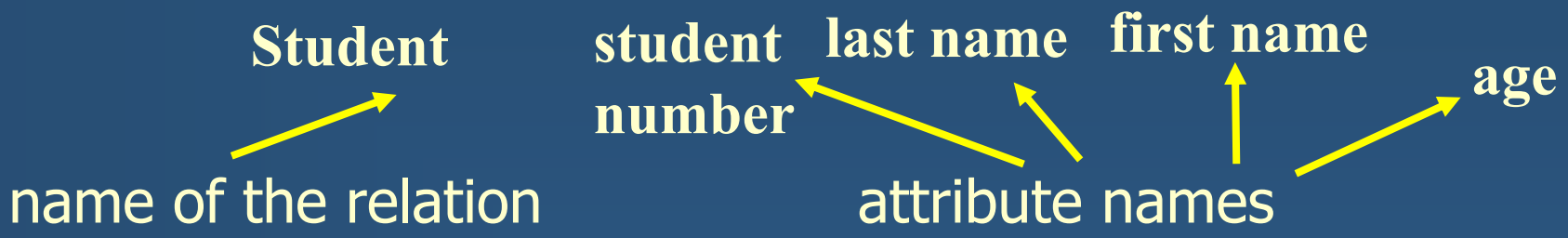
# Conceptual Modeling

*(Part 2)*

# The Relational Model

- a logical model is very simple
- defined by Ted Codd in 1970; Turing prize in 1986. Developed by IBM Labs.
- used today by a larger portion of commercial DBMS (Oracle, Informix, DB2, Ingres, Sybase, dBase, Access …) et GIS
- the model has two concepts:
  - relation (table)
  - attribute (column)

# Basic Concepts




Logical Schema

| Student | | | |
|---|---|---|---|
| S. Nu. | last name | first name | age |
| 136 | Dupont | Jean | 19 |
| 253 | Aubry | Annie | 20 |
| 101 | Duval | André | 21 |
| 147 | Dupont | Marc | 21 |

population

tuple or record

# Relational Schema

- a database is a set of relations
- the schema of a relational database = a set of relational schemas: R1, R2, …, Rx
- a relational schema = a set of attributes

  Ri = (Attribute1/domain1, A2/d2, …, Ay/dy)

  or, more simply,

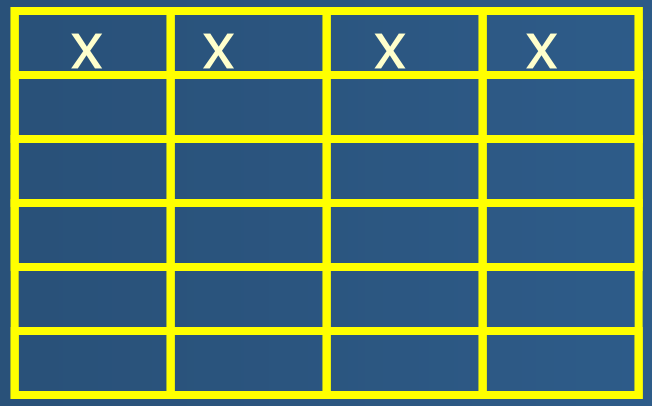
  Ri = (A1, A2, …, Ay)

# Structuring Rules

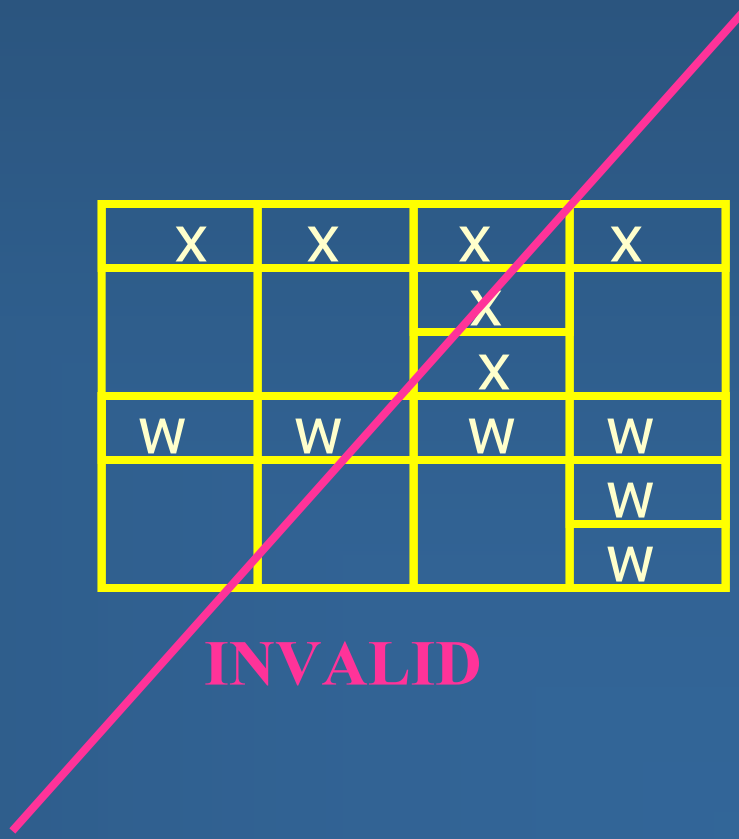
attributes: simples and single values
(a domain of simple values)

regular flat structure

tuple




x: one and only one
value per attribute



# Null Values

- An attribute might not be valued for a tuple: we call these values a 'null value'.
  - Example: we don't know Annie's Age, nor Duval's first name.

| | | | |
|---|---|---|---|
| 136 | Dupont | Jean | 19 |
| 253 | Aubry | Annie | NULL |
| 101 | Duval | NULL | 21 |
| 147 | Dupont | Marc | 21 |

# Rules regarding the Identifier

- All relations have an Identifier (key)
  - there cannot be two identifiers in the same relation
- An Identifier cannot take a null value

| Student | S. Numb. | last name | first name | age |
|---|---|---|---|---|
| | 136 | Dupont | Jean | 19 |
| | 253 | Aubry | Annie | 20 |
| | 101 | Duval | André | 21 |
| | 147 | Dupont | Marc | 21 |

# External Identifiers

Course (NameC, schedule, prof)

| | | |
|---|---|---|
| BD | Mercredi 15-17 | Duval |
| SE | Mardi 16-19 | Malin |

Enrolled (S. Numb., NameC)

| | |
|---|---|
| 253 | SE |
| 136 | BD |
| 253 | BD |
| 101 | SE |

**Enrolled** converts a RT from Student to Course. It stores the identifiers of Student and Course. Course.NameC is an external identifier on Course.

# Value Domains

- An attribute's Domain is the set of simple values which are allowed in the attribute.
- Examples:
  - Dname : sequence of characters with mamimum length of 30
  - Dnum : whole numbers between 0 and 99999
  - Dcolor : {"blue", "green", "yellow"}
  - Dage : whole numbers between 16 and 65

# Definition of a Relation

- A Relation is defined by:
  - its name
  - its set of tuples < Attribute name : domain >
  - its identifier(s)
  - its semantic definition (in English)
- Example :
  - Student (St.Numb : Dnum, FirstName : Dname, LastName : Dname, Age : Dage)
  - Identifier : St.Numb
  - Definition: any student currently registered at ISI

# Modeling Constraints

The notion of a complex or multivalued attribute do not exist in the relational model. We must model it a different way.

For a complex attribute, you must choose between the compound or the components.

For a multivalued attribute, you must create another relationship (one for each multivalued attribute).

# Representing Complex Attributes

- If Address : street name, number, city, PC
- Solution 1:
  - an Attribute expressed by its components:

    street name, number, city, PC

    “Rue de Bourg”,”2”,”Lausanne”,”1003”
  - it is possible to define a perspective that restores the global notion of address
- Solution 2:
  - an Attribute expressed by a string (a chain of characters):

    "Rue de Bourg 2  Lausanne  1003"

# Representation of a Multivalued Attribute

Example: store the name of students

INCORRECT:

Several attributes: Name1,Name2...

CORRECT: create an additional relation

StudNames (#N° Stud, Name )

| | |
|---|---|
| 136 | Jean |
| 136 | Marie |
| 101 | André |
| 253 | Annie |
| 253 | Claudine |

Or ordered list:

StudNames2 (#N° Stud, N°Name, Name)

# The Identifier of a Relation

- A relation can have several identifiers

  StudNames2 (#N°Stud, N°Name, Name)

- Definition: a relation's identifier is a minimal set of of the relation's attributes, such that there are no 2 tuples with the same identifier values.
- Rule : all of an identifier's attributes must be a non-null values.

# External Identifiers

- Describe the links between relations
- Enrolled (#N°Stud : Dnum, #NameC: Dnom)

  N°Stud **references a** Student

  NameC **references a** Course
- If the referenced relation possesses several identifiers, we need to specify:

  N°Stud **references a** Student.N°Stud
- Referential Integrity is ensured by the DBMS: the external identifiers necessarily refer to existing tuples.

# Summary

- A relational schema is composed of :
  - for each relation :
    - the name of the relation
    - definition
    - attributes + domains
    - identifier(s)
    - possibly external identifier(s)
    - associated integrity constraints
  - and other integrity constraints that relate to multiple relations.

# Example of a Relational Schema

- Domains:
  - Dname: sequence of characters (30 or more)
  - Dch100 : sequence of characters (100 or more)
  - Dyear : [1970 : 2020]
  - Dmark : [0.0 : 20.0 ]
  - Ddate : [1 :31 ] / [1 :12 ] / [1920 :2020]

# Example of a Relational Schema

- **Relation : *Person***

  **Attributes:** *n°P* : non-null whole numbers
  *name : Dname* non- null
  *adr* : *Dch100* non-null

  **Identifier** : (*n°P* )

  **Definition** : *all students and all teachers at the school (currently)*

- **Relation : *PersonName***

  **Attribute :** *n°P* : non-null whole numbers
  *name* : *Dname* non-null

  **Identifier** : (*n°P* + *name* )

  **External Identifier** : *n°P* references a *Person*

  **Definition** : *peoples' names*

# Example of a Relational Schema

- **Relation : *Student***

  **Attributes:** *n°P* : non-null whole number
  *n°S* : non-null whole number
  *dateN* : *Ddate* non-null

  **Identifier :** (*n°S*)
  (*n°P*)

  **External Identifier :** *n°P* references a *Person*

  **Definition :** *Any individual who is currently enrolled, or who has already graduated.*

- **Relation : *StudentProgram***

  **Attributes:** *n°S* : non-null whole number
  *year* : *Dyear* non-null
  *program*: *Dname* non-null

  **Identifier :** (*n°S* + *program*)

  **External Identifier :** *n°E* references a *Student.n°S*

  **Definition :** *any program studied by students*

# Example of a Relational Schema

- **Relation : *Teacher***

  **Attributes:** *n°P*: non-null whole number
  *n°Tea*: non-null whole number
  *ph*: non-null whole number
  *status*: *Dname* non-null
  *bank* : *Dname* non-null
  *agency*: *Dname* non-null
  *accnt* : non-null whole number

  **Identifier** : (*n°Tea*)
  **External Identifier** : *n°P* references a *Person*
  **Definition** : *any individual who are teaching one or more courses at the school.*

- **Relation : *Course***

  **Attributes:** *nameC*: *Dname* non-null
  *cycle* : non-null whole number
  *n°Tea*: non-null whole number

  **Identifier** : (*nameC*)
  **External Identifier** : *n°Tea* references a *Teacher*
  **Definition** : *any course currently offered by the school.*

# Example of a Relational Schema

- **Relation : *Pass***

  **Attributes:** *n°S* : non-null whole number
  *nameC*: *Dname* non-null
  *grade*: *Dgrade* non-null
  *year* : *Dyear* non-null

  **Indentifier** : (*n°E* + *nameC* )

  **External Identifiers** : *n°S* references a *Student.n°S*
  *nameC* references a *Course*

  **Definition** : *the student n°S has passed the Course nameC with this grade*

- **Relation : *Enrolled***

  **Attributes:** *n°S* : non-null whole number
  *nameC*: *Dname* non-null

  **Identifier** : (*n°S* + *nameC*)

  **External Identifier** : *n°S* references a *Student.n °S*
  *nameC* references a *Course*

  **Definition** : *currently, the student n°S is enrolled in Course nameC*

# Example of a Relational Model

- **Relation : *Prerequisite***

  **Attributes:** *nameC*: *Dname* non-null

  *nameCprereq* : *Dname* non-null

  **Identifier** : (*nameC* + *nameCprereq* )

  **External Identifier** : *nameC* references a *Course*

  *nameCprereq* references a *Course*

  **Definition** : *the course nameCprereq is a prerequisite of the course nameC*

  **Intergrity Constraint** : *in all tuples, nameCprereq must be different than nameC*