ICT & Business Intelligence & CRM
Logical Model

Andrea Vandin
Scuola Superiore Sant'Anna
Paolo Ferragina
Scuola Superiore Sant'Anna
Anna Monreale
Università di Pisa

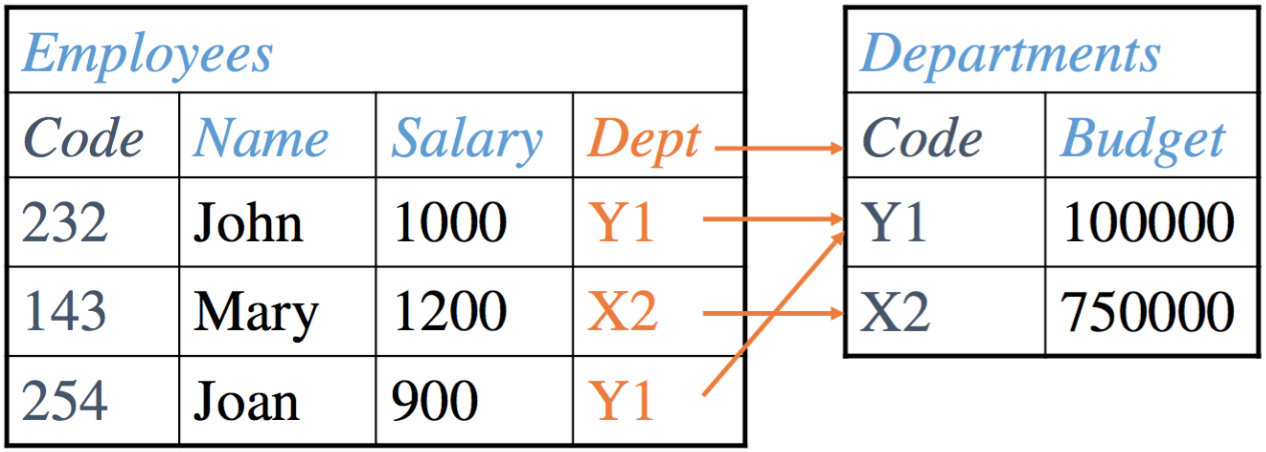
Reference: Chapter 3.3 of Databases Essentials, Antonio Albano

Phases for DB realization

- User requirements analysis & specification
  - collecting **user needs** and normalizing them according to standards
- Conceptual design (PREVIOUS CLASS)
  - Focus is more on how tables are related with each other
    - e.g., we do not need to consider all attributes/domains of tables
  - This is the phase in which requirements are formalized and integrated into a **global conceptual schema**
    - (Global because it considers all tables)
  - We use a DBMS-independent (conceptual) language
- **Logical design (THIS CLASS)**
  - The conceptual schema is mapped into a **logical schema**
    - We use the data model supported by the chosen DBMS
    - We get closer to the actual DB creation. It depends on the chosen DBMS
    - Logical schema describes a relational DB
- Physical design
  - concerns the selection of the **data structures** used to store and retrieve the data.

Relational Data Model

- We generate the Logical Model by following a systematic process
  - organize all data in flat **tables** of **rows** with a primary key
  - transform associations as **foreign keys**



Employees			
Code	*Name*	*Salary*	*Dept*
232	John	1000	Y1
143	Mary	1200	X2
254	Joan	900	Y1

Departments	
Code	*Budget*
Y1	100000
X2	750000

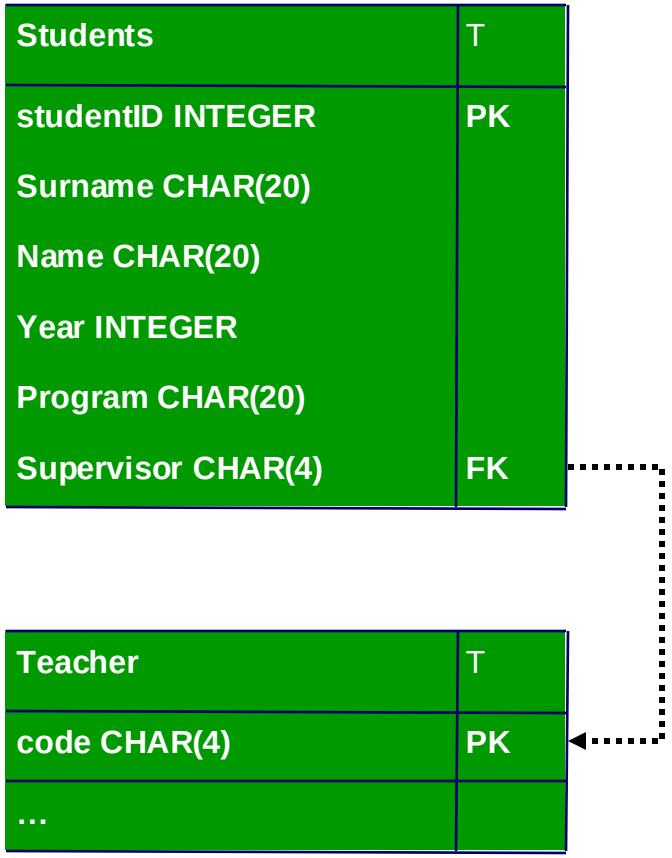
Algorithm for the Logical Design

- **Step I**: Translate all classes not involved in hierarchies
- **Step II**: Translate all hierarchies
- **Step III**: Translate multivalued attributes into tables
- **Step IV** : Translate N-N relationships
- **Step V** : Translate 1-N relationships
- **Step VI** : Translate 1-1 relationships
- **Step VII** : Add other possible constraints

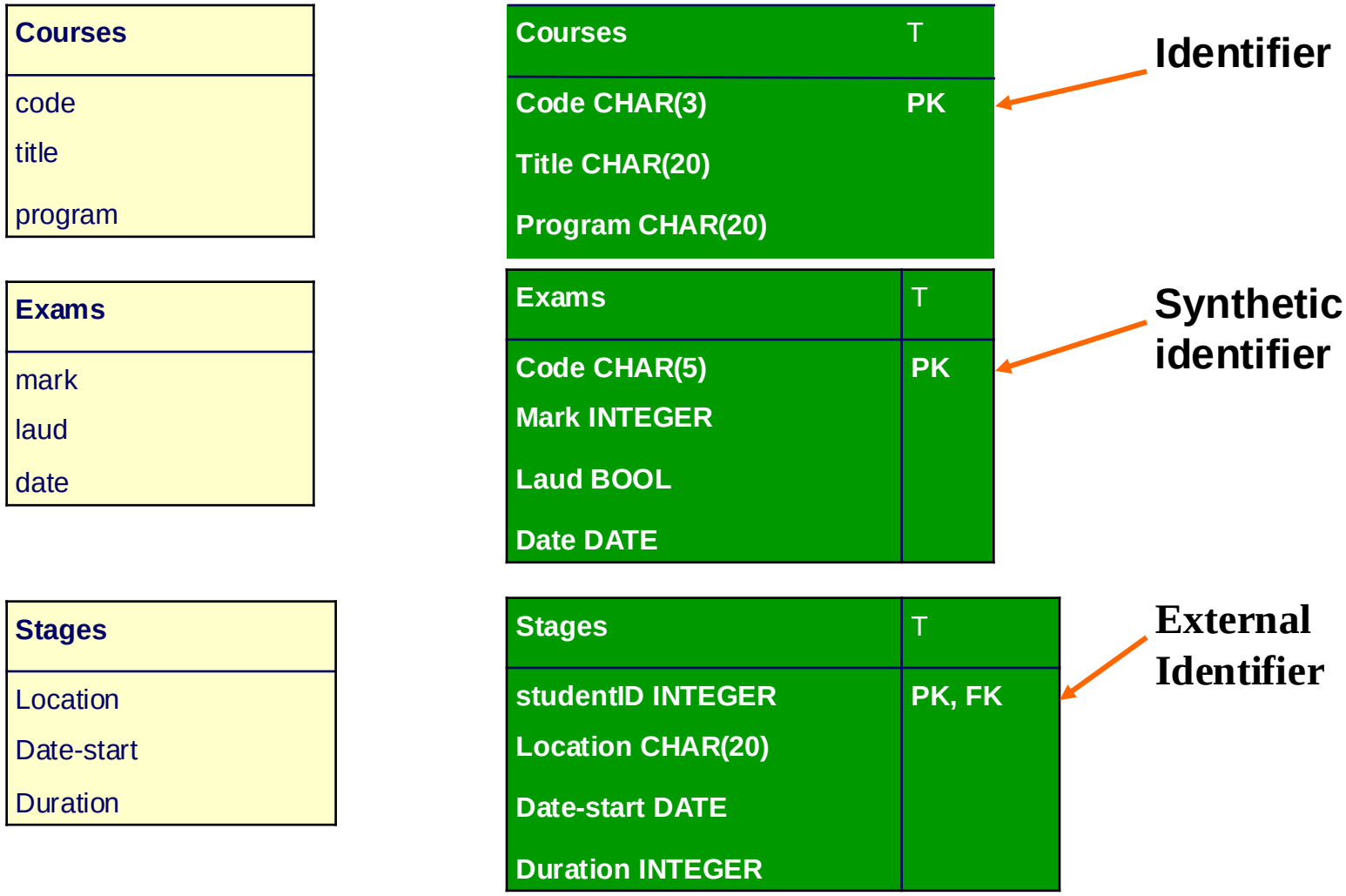
Step I: Translate classes not in hierarchies

- Classes become **tables** containing
  - Attributes
  - primary key
  - foreign keys

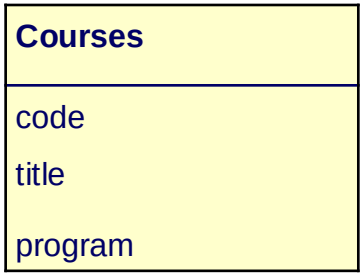
- How to choose a **Primary key**?
  - Simple to be used and compact
  - We can use sets of attributes if compact
  - We can use a synthetic identifier



Step I: Translate classes not in hierarchies



Step I: Translate classes not in hierarchies



Courses	T
Code CHAR(3)	PK
Title CHAR(20)	
Program CHAR(20)	

Code	Title	Program
PR1	Programming	Bachelor
ASD	Algorithms	Bachelor
DB1	Databases	Master

Step I: Translate classes not in hierarchies

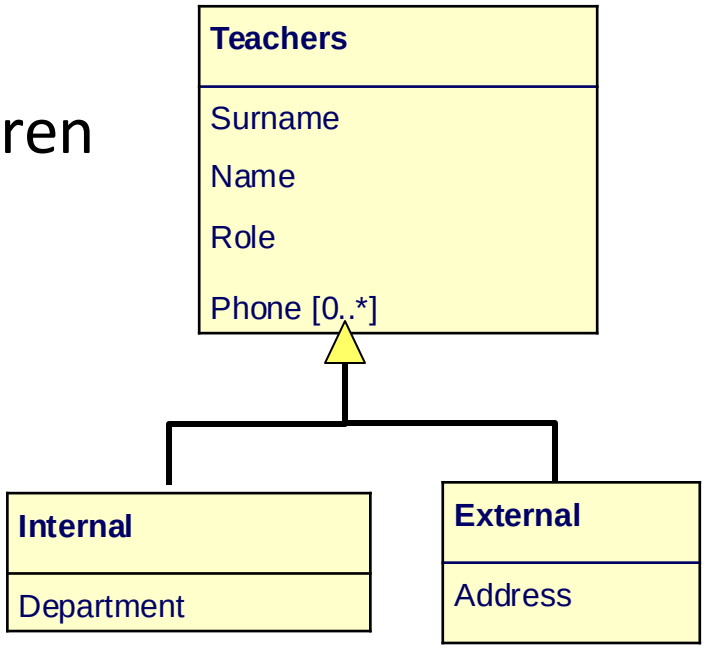
Stages
location Date-start duration

Stages	T
studentID INTEGER Location CHAR(20) Date-start DATE Duration INTEGER	PK, FK

StudentID	Location	Date-start	Duration
444	Microsoft	2002-05-15	3
77777	Microsoft	2002-05-15	3
88888	Basica	2002-09-01	3

Step II: Translate all hierarchies

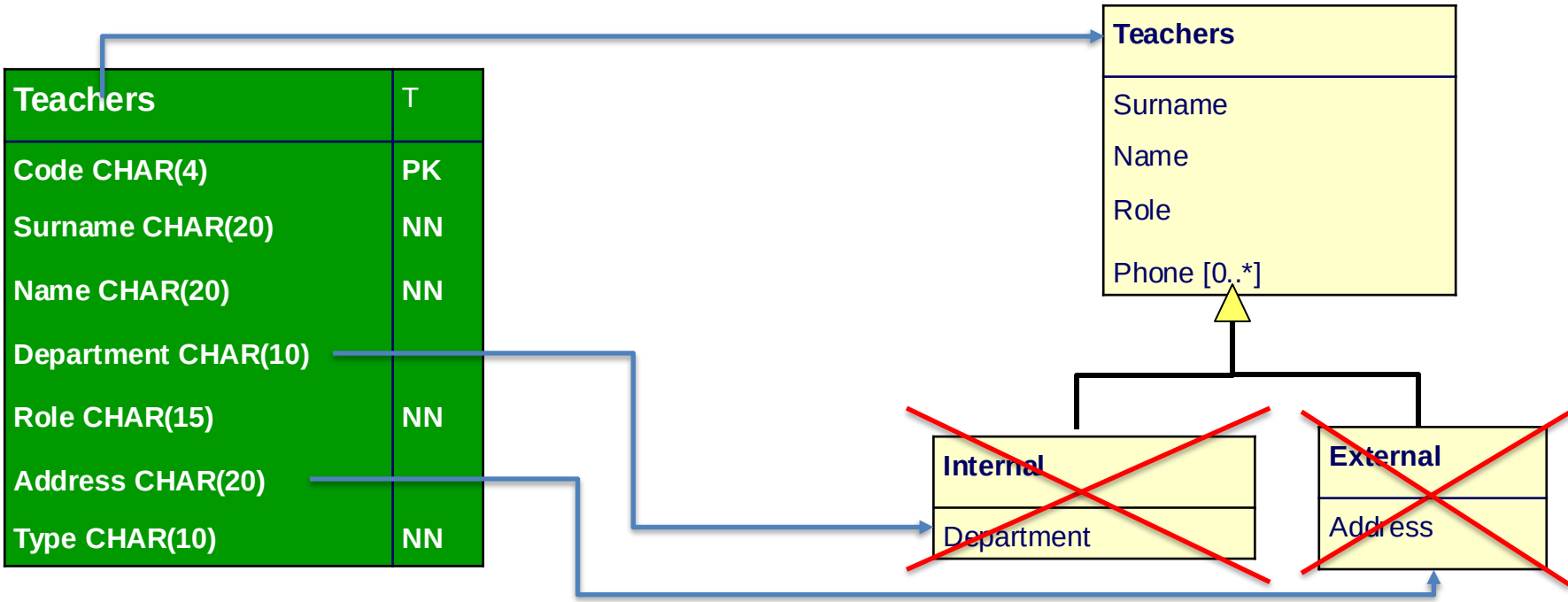
- Relational Data Model **does not** have Hierarchies
- We need to get rid of them with tables
- Three possible cases
  1. Translate only the parent class
  2. Translate only children
  3. Translate both parent and children



Step II: Translate all hierarchies

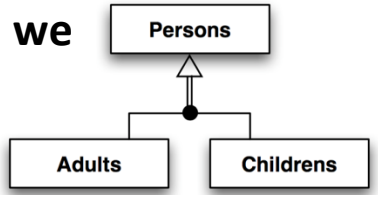
1 Translate only the parent class

- One table with the name of the parent class
- The table contains the attributes of both parent and children
- Add an attribute **TYPE** to distringuish the instance
- Generation of **NULL values**
- It is convenient if the children **are not used so much** in the application



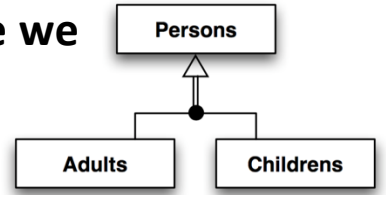
Step II: Translate all hierarchies

- **Translate only children classes**
  - One table for each child
  - Each child table inherits both attributes and relationships of the parent
  - Possible only when the the **hierarchy is complete. Otherwise we**
    - **Cannot represent elements not in sub-classes**
    - **Do not know where to add an element belonging to both subclasses**
  - It is convenient if the application **uses often the children**



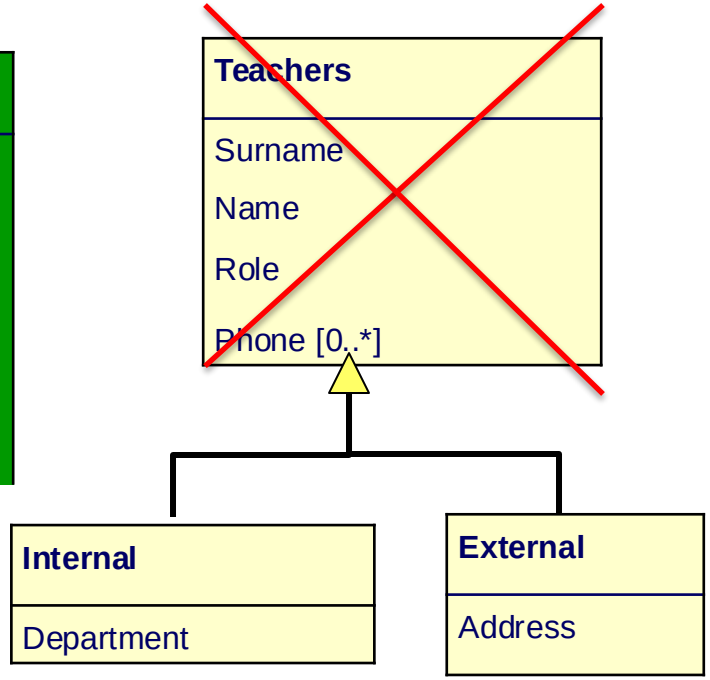
Step II: Translate all hierarchies

- **Translate only children classes**
  - One table for each child
  - Each child table inherits both attributes and relationships of the parent
  - Possible only when the the **hierarchy is complete. Otherwise we**
    - **Cannot represent elements not in sub-classes**
    - **Do not know where to add an element belonging to both subclasses**
  - It is convenient if the application **uses often the children**



InternalTeachers	T
Code CHAR(4)	PK
Surname CHAR(20)	
Name CHAR(20)	
Department CHAR(10)	
Role CHAR(15)	

ExternalTeachers	T
Code CHAR(4)	PK
Surname CHAR(20)	
Name CHAR(20)	
Address CHAR(10)	
Role CHAR(15)	



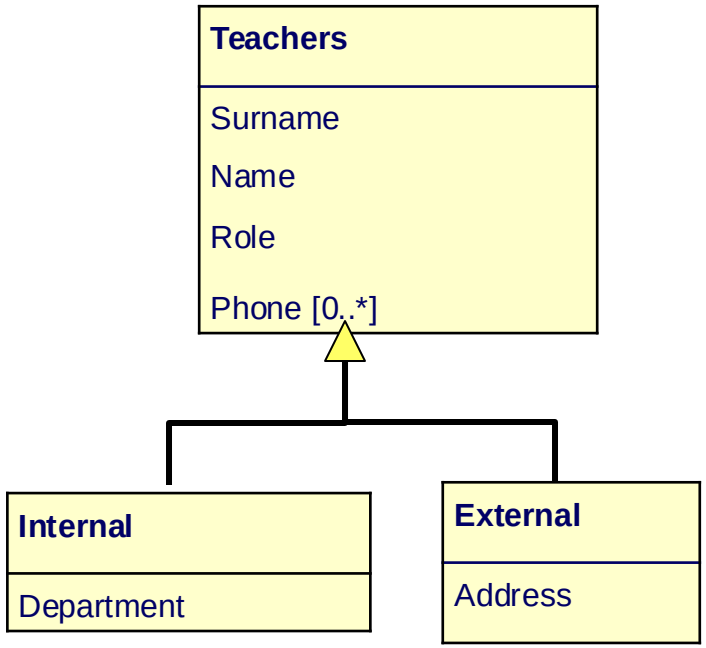
Step II: Translate all hierarchies

Teachers	T
Code CHAR(4)	PK
Surname CHAR(20)	
Name CHAR(20)	
Role CHAR(15)	

ExternalTeachers	T
Code CHAR(4)	PK,FK
Address CHAR(10)	

InternalTeachers	T
Code CHAR(4)	PK,FK
Department CHAR(10)	

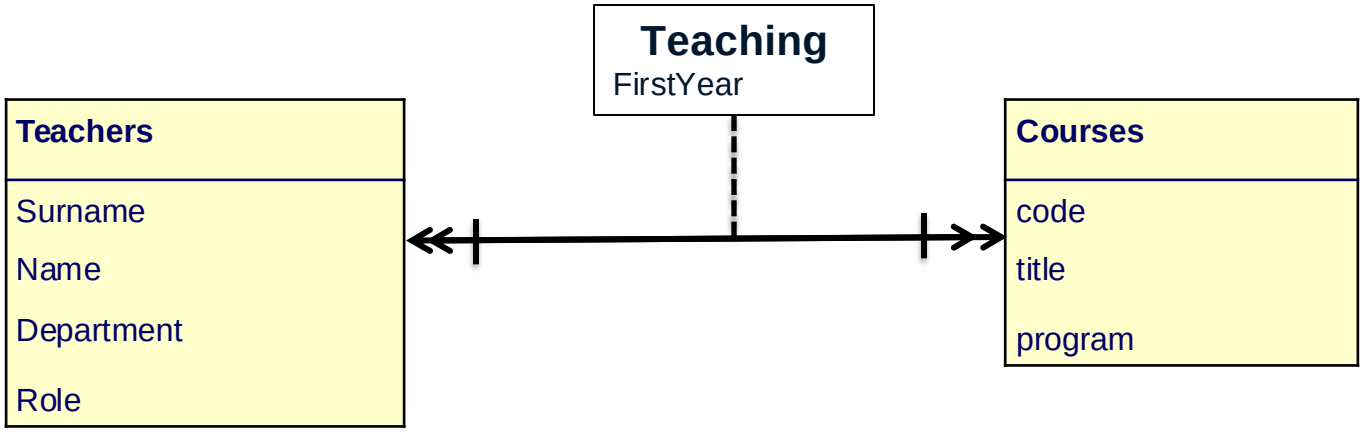
- **Translate both parent and children**
  - One table per child, one for the parent
  - Each class has its attributes
    - Good: No missing values, No redundancies
    - Bad: Might lead to slower performances
  - It is necessary if the application **uses often both children and parent**

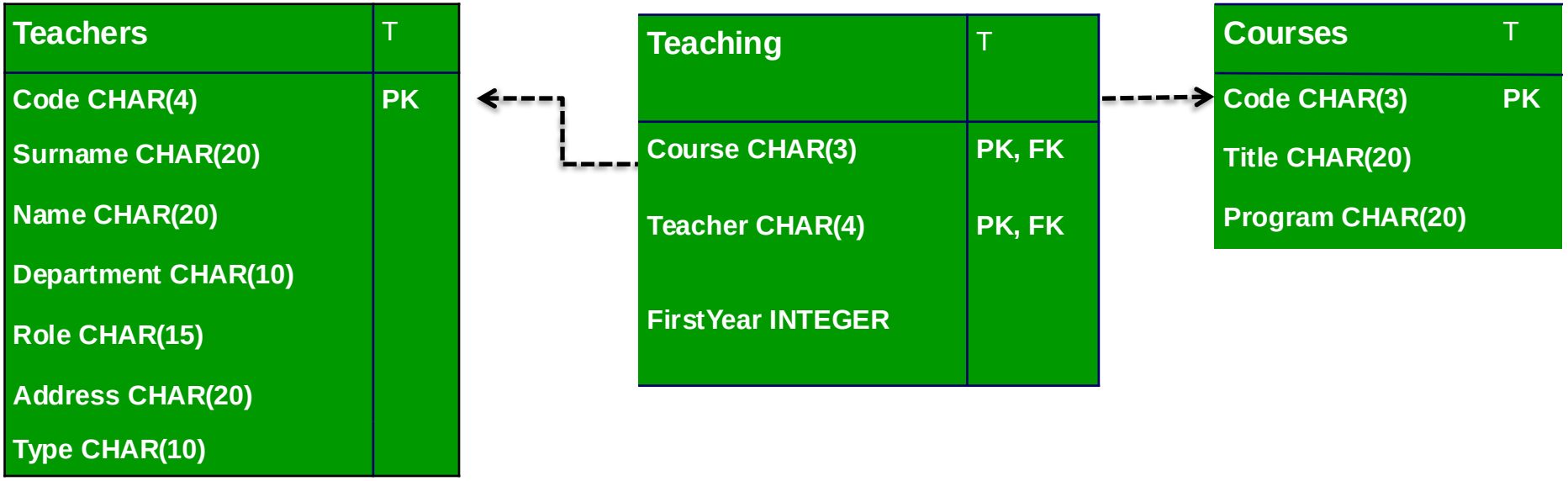


Step III: Translate multivalued attributes

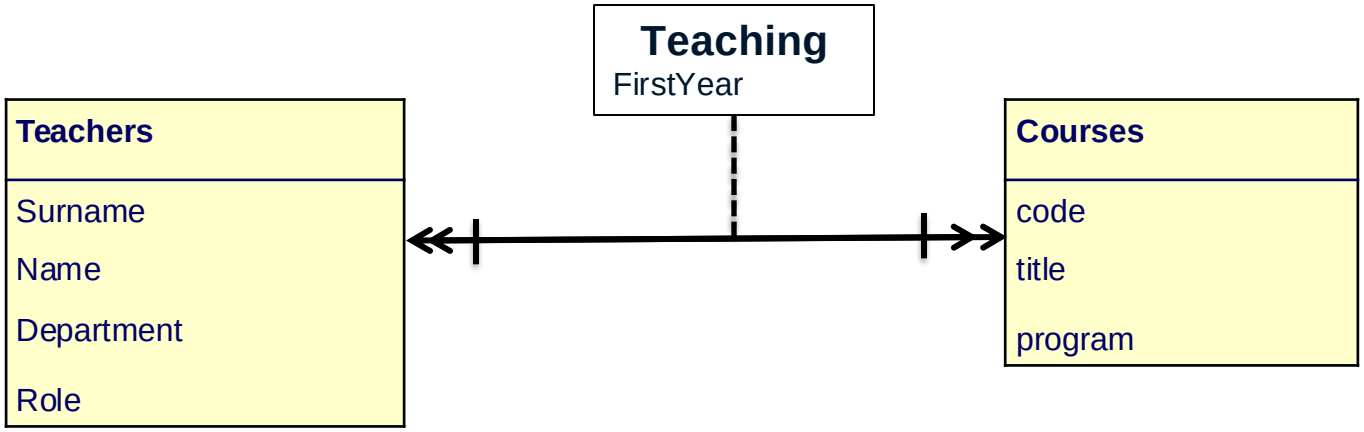
- Each multivalued attribute becomes a new table
  - E.g., a list of phone numbers
- The new table contains a foreign key towards the original class

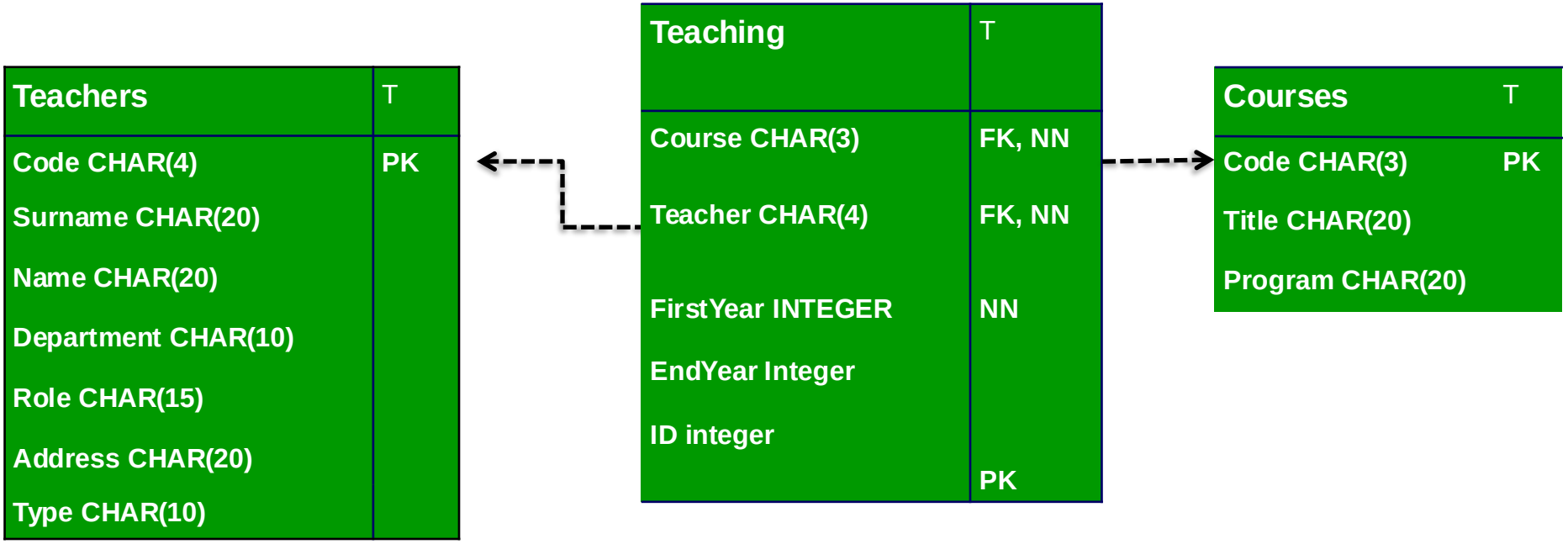
Step IV : Translate N-N relationships





Step IV : Translate N-N relationships





Step IV : Translate N-N relationships

Teachers

codice	cognome	nome	…
FT	Pedreschi	Dino	…
CV	Monreale	Anna	…
ADP	Giannotti	Fosca	…

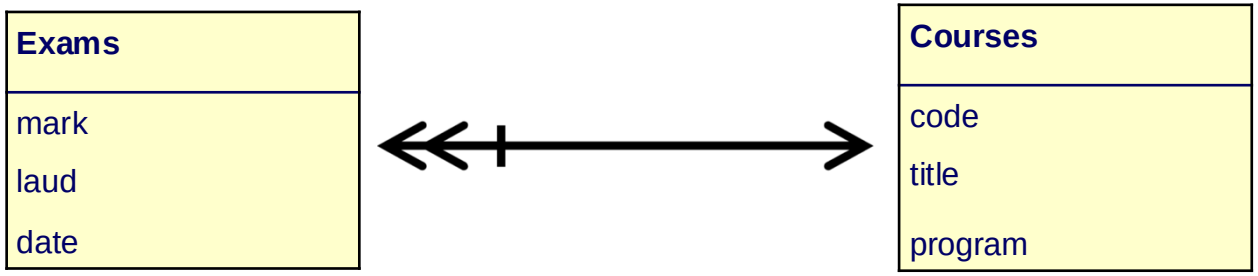
Teaching

docente	corso	primoAnnoTit
FT	PR1	2001
CV	ASD	2002
FT	ASD	1999
…	…	

Courses

Code	Title	Program
PR1	Programming	Bachelor
ASD	Algorithms	Bachelor
DB1	Databases	Master

Step V : Translate 1-N relationships



Exams	T
Code CHAR(5)	PK
Course CHAR(3)	FK
Mark INTEGER	
Laud BOOL	
Date DATE	

Courses	T
Code CHAR(3)	PK
Title CHAR(20)	
Program CHAR(20)	

Courses

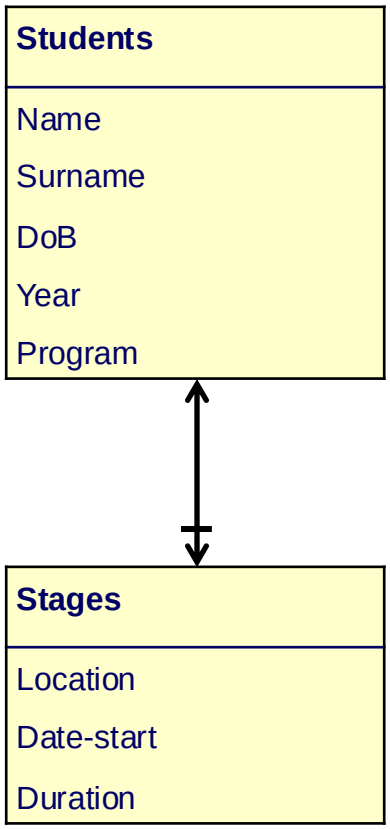
Code	Title	Program
PR1	Programming	Bachelor
ASD	Algorithms	Bachelor
DB1	Databases	Master

Exams

Code	Course	Mark	Laud	Date
pr101	PR1	27	false	2002-06-12
asd01	ASD	30	true	2001-12-03
BD101	INFT	24	false	2001-09-30
pr102	PR1	21	false	2002-06-12
asd02	ASD	20	false	2001-12-03
asd03	ASD	28	false	2002-06-13

Step VI : Translate 1-1 relationships

- Similar to 1-N. But we can choose where to put the foreign key
- It is better in the class where we have total relationship



Students	T
studentID INTEGER	PK
Surname CHAR(20)	
Name CHAR(20)	
Year INTEGER	
Program CHAR(20)	
Supervisor CHAR(4)	FK

Stages	T
studentID INTEGER	PK, FK
Location CHAR(20)	
Date-start DATE	
Duration INTEGER	

Step VII: Add other possible constraints

- Now we have:
  - Tables
  - Attributes
  - Primary keys
  - Foreign keys
- We need to add other constraints
  - NOT NULL
  - DEFAULT
  - CASCADE
  - CHECK
  - …..