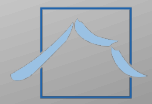
Informatics II for Engineering Sciences (MSE)

Chapter I – Databases: Schema

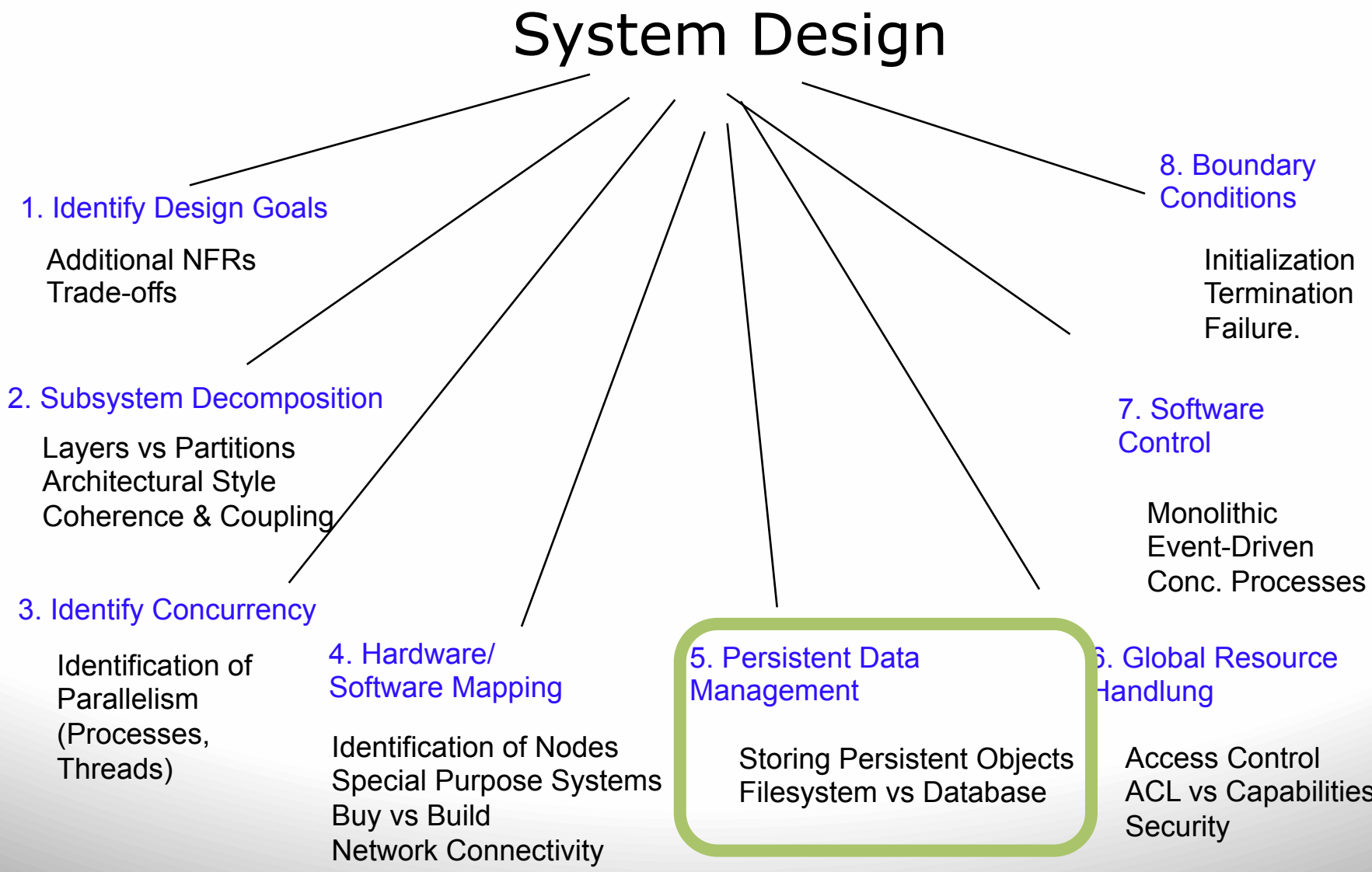
Bernd Brügge Ph.D. , Jan Knobloch, Emitzá Guzmán

Outline

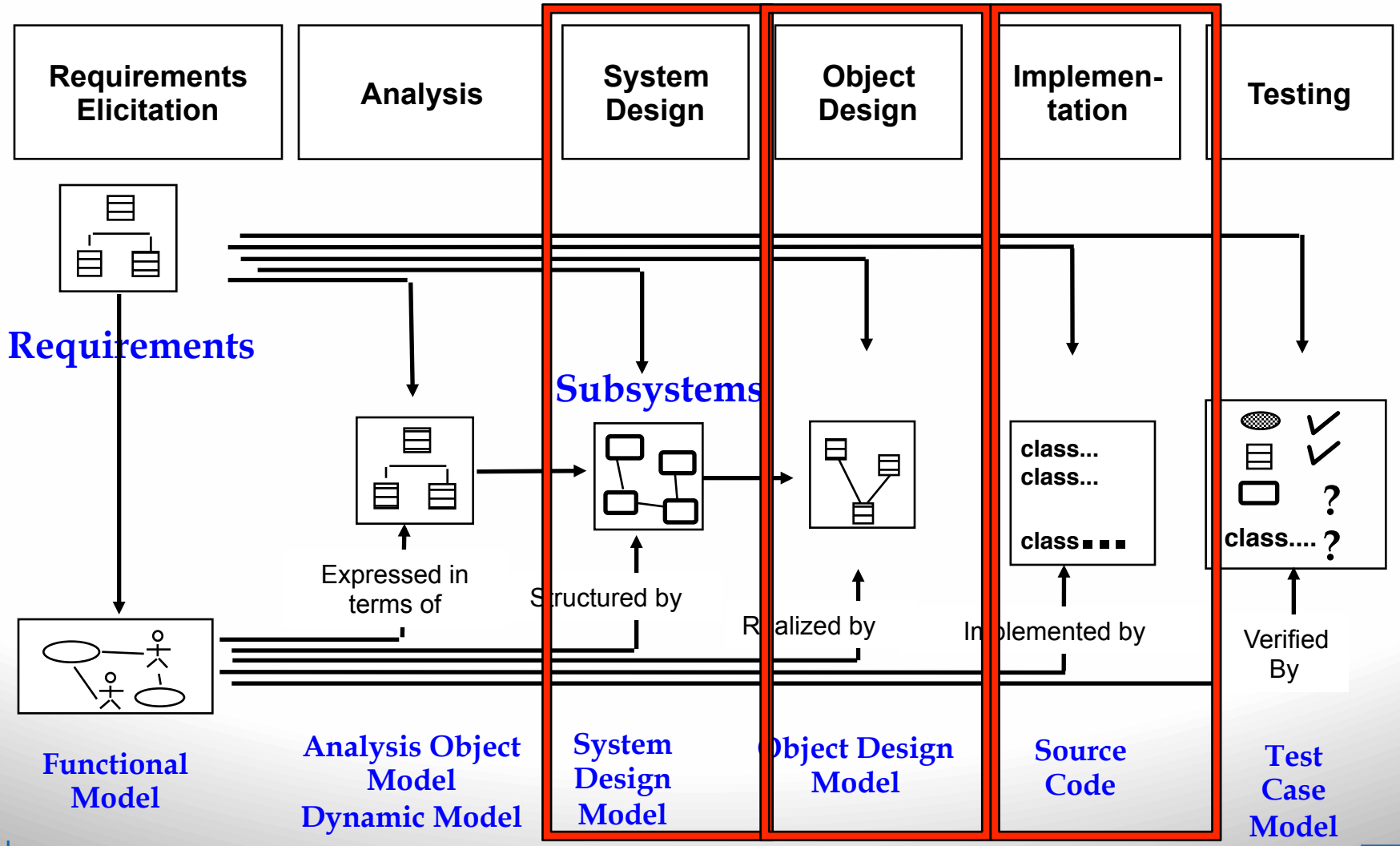
- Intro
- Relational Database
- SQL
- ORM



Modeling in System Design



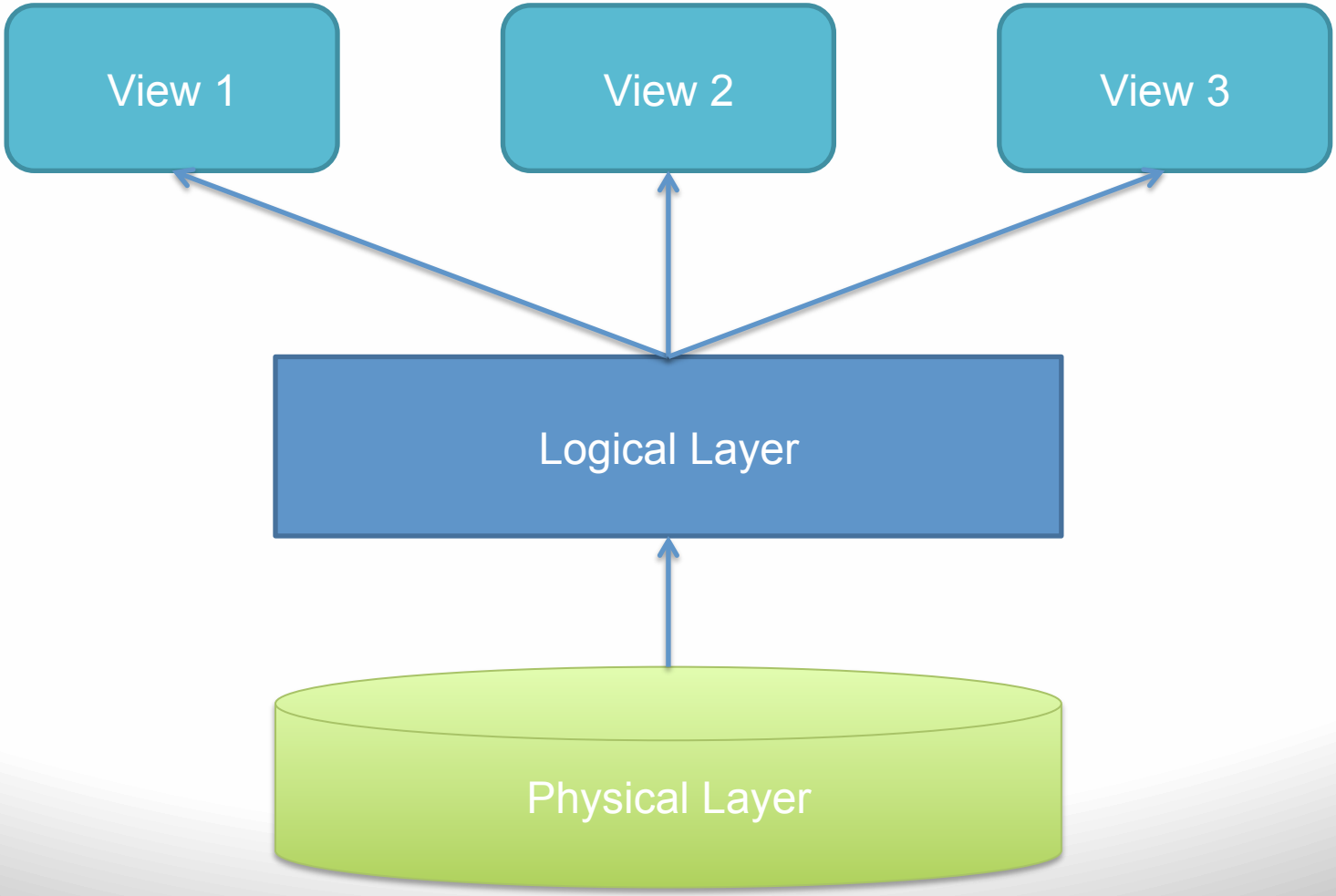
Position in the Software Lifecycle



Motivation – Databases

- Allow concurrent data access
- Avoid redundancy and inconsistency
- Rich access to data
- Avoid loss of data
- Enforce integrity rules
- Ensure security and privacy

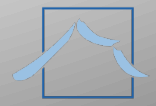
Architecture of Database Systems



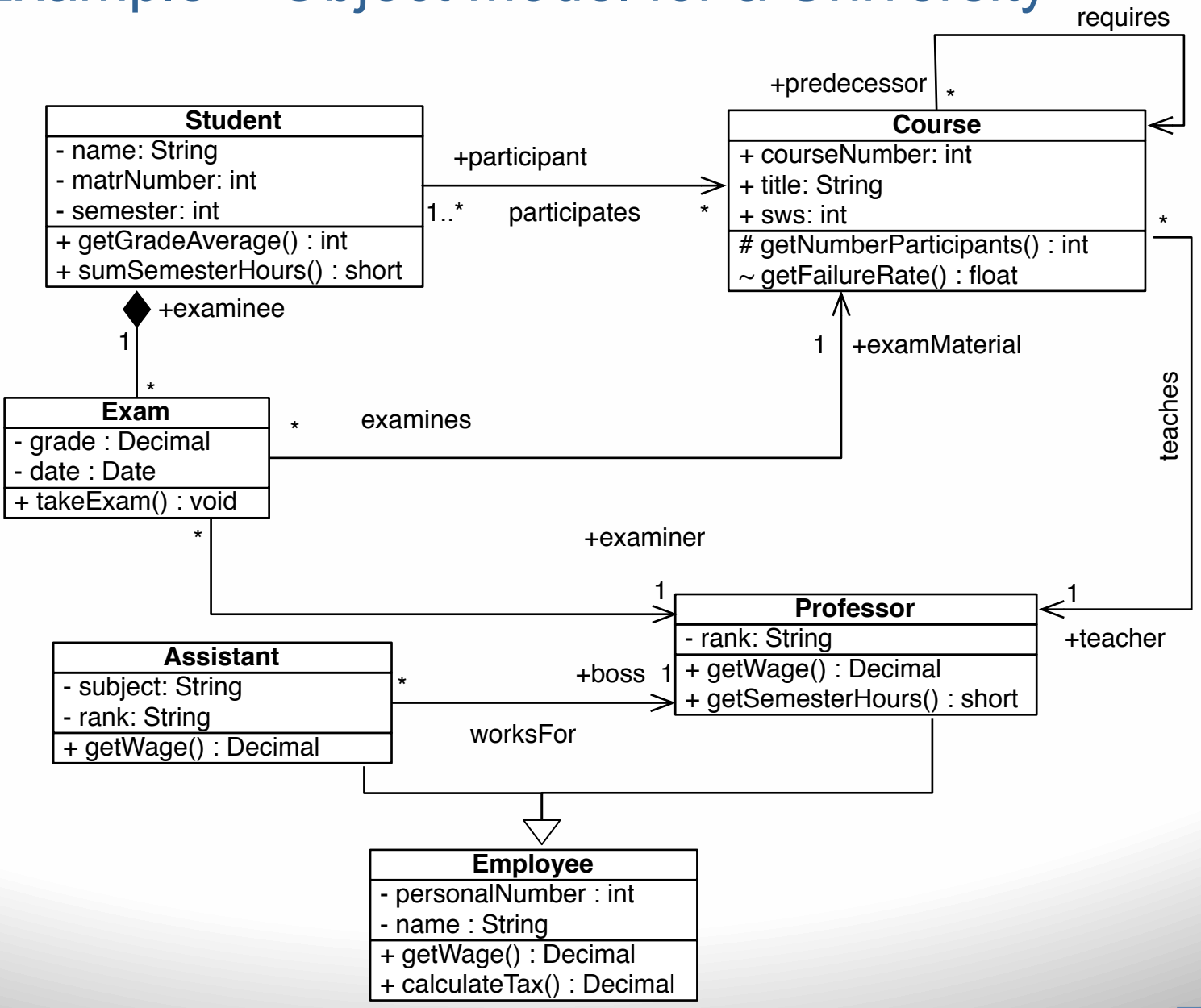
From Object Model to Relational Model

Mapping an Object Model to a Database

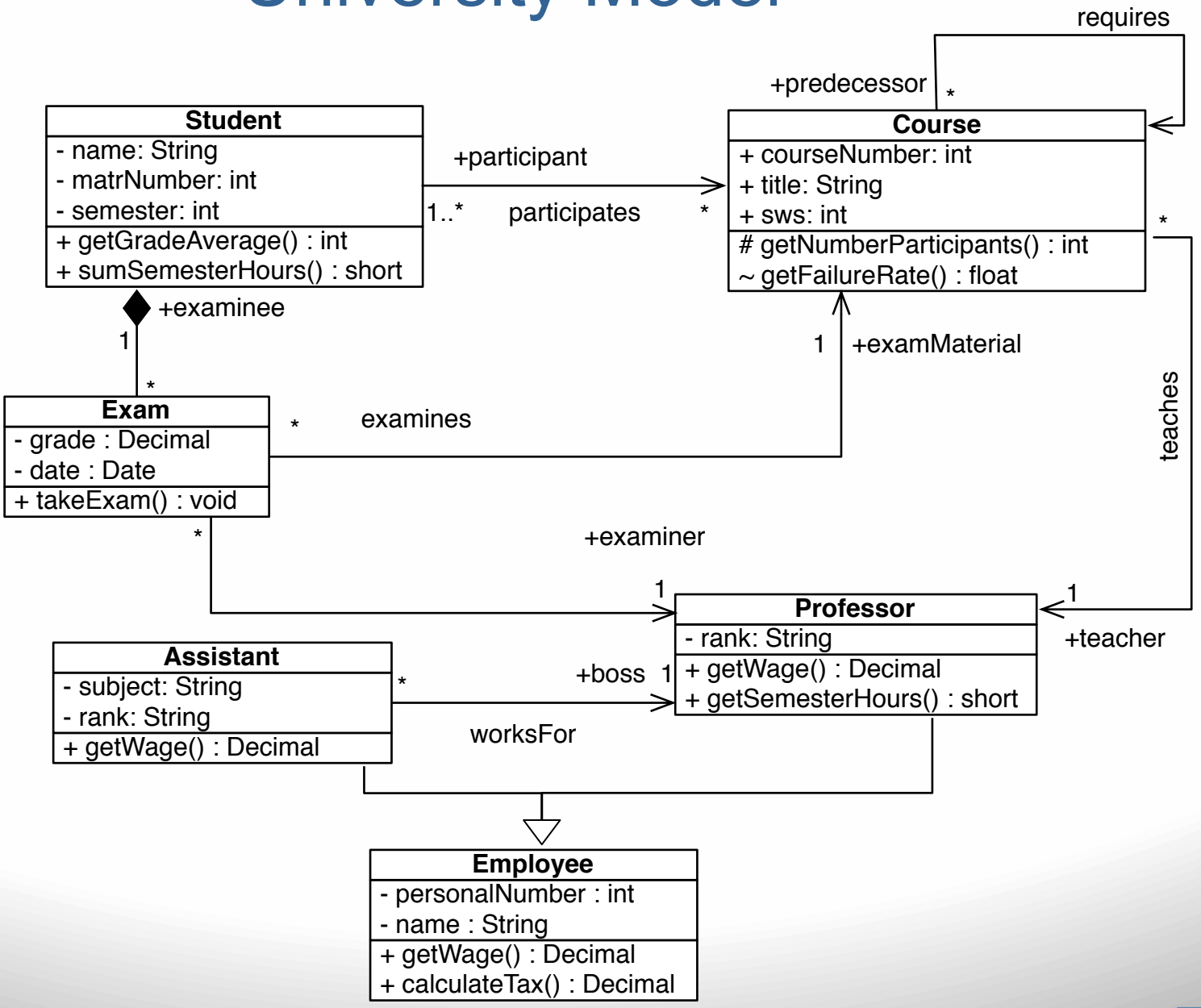
- UML object models can be mapped to relational databases:
  - Some degradation occurs because all UML constructs must be mapped to a single relational database construct - the **table**
- Mapping of classes, attributes and associations
  - Each *class* is mapped to a **Table**
  - Each class *attribute* is mapped onto a column in the table
    - The collection of all attributes is called the **Schema**
  - An *instance* of a class represents a row in the table (**Tuple**)
  - A *one-to-many association* is implemented as buried foreign key
  - *A many-to-many association* is mapped into its own table
- Methods are not mapped.



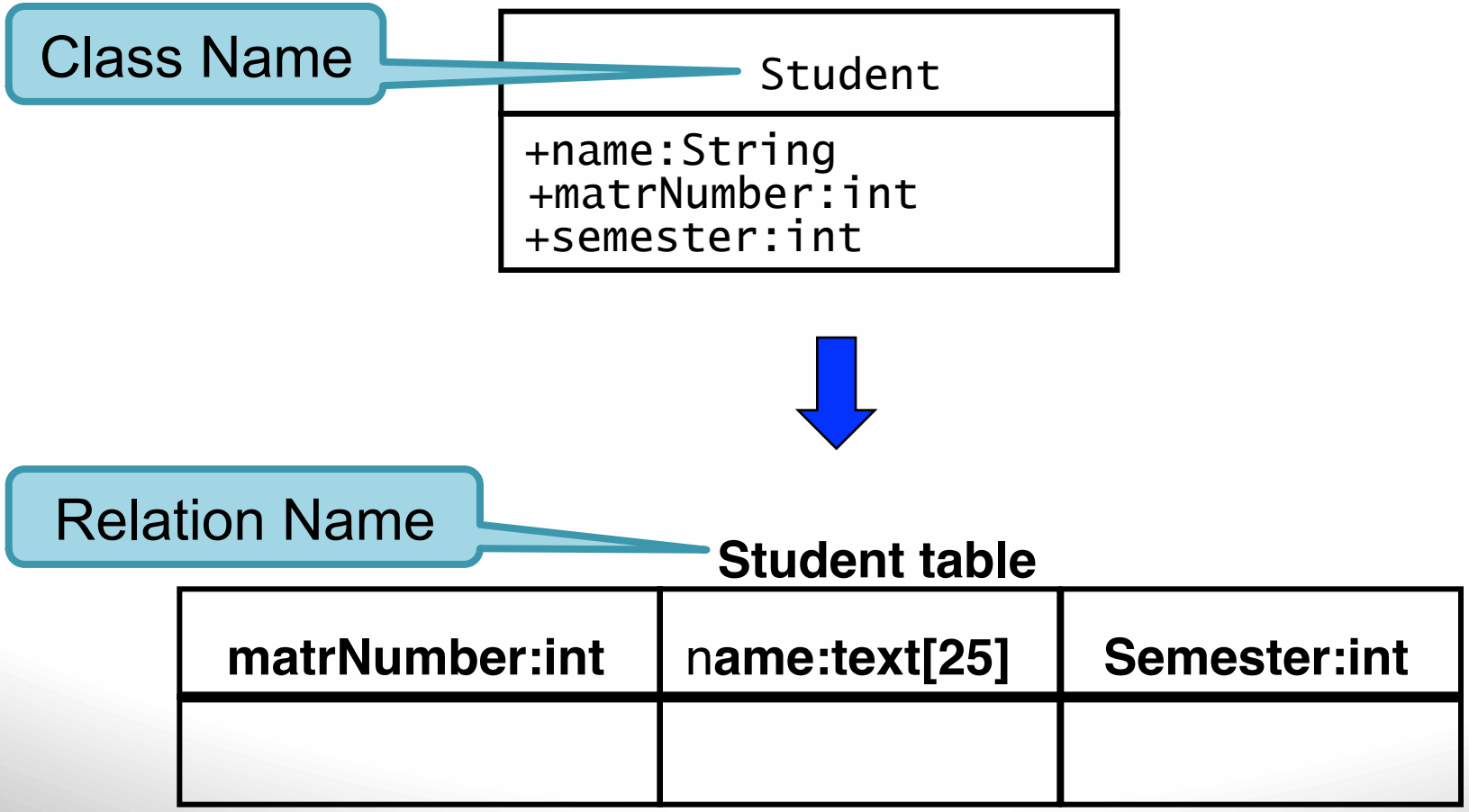
Example – Object Model for a University



University Model



Mapping a Class to a Table



Relational Model

Relation name

Student Table

Class Name

Attribute

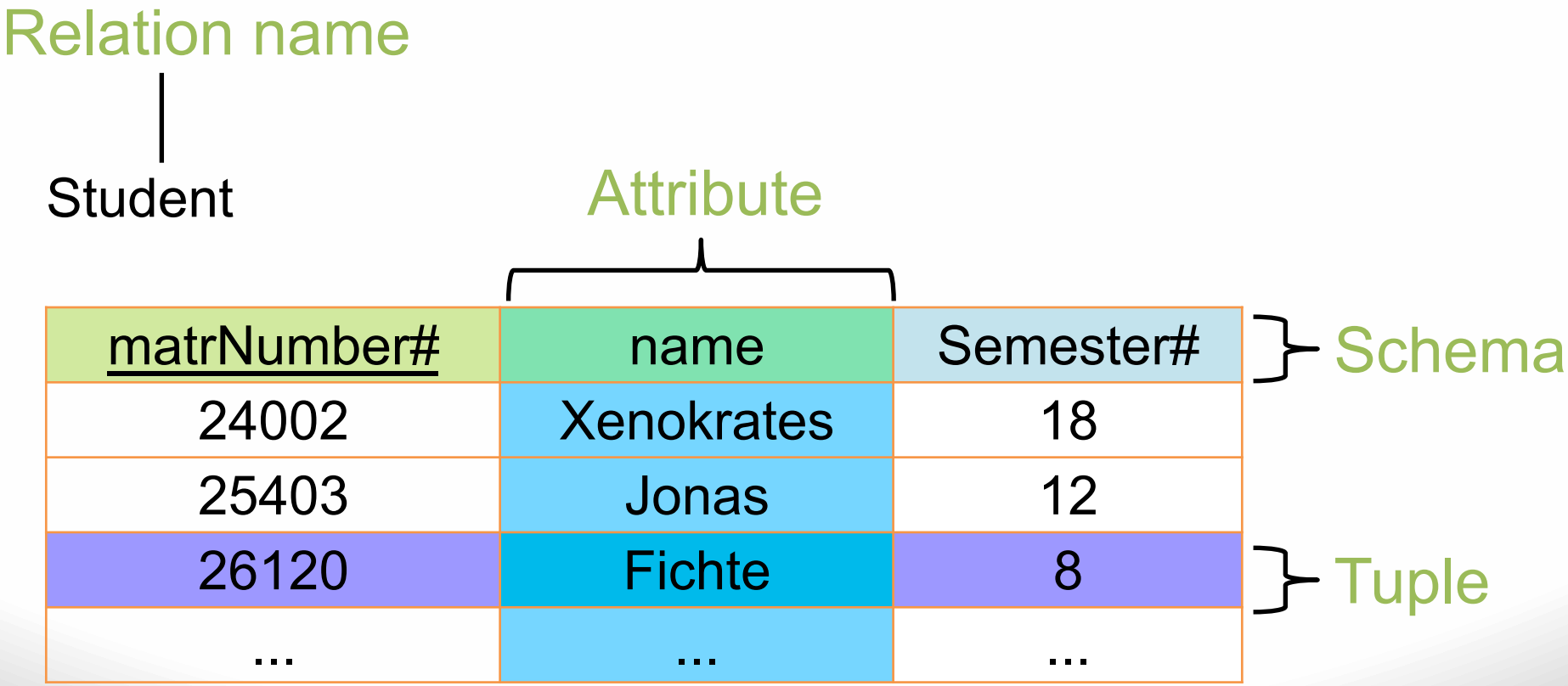
matrNumber#	name	Semester#
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	8
...	...	...

Schema

Tuple

Class Instance

Relational Model



Student: {[matriculationNr: int, name: String, semester: int]}

Relational Model : Mathematical Description

- **Domain D (span for possible values)**
  e.g. string, int, 1-10
- **Relation: R $\subseteq$ *D1 x D2 x ... x Dn***
  eg.: phoneBook $\subseteq$ *string x string x integer*
- ***Tuple: t $\in$ R***
  eg: („Mickey Mouse“, „Main Street“, 4711)
- ***Schema: structure of stored data***
  eg: phoneBook: {[Name: string, address: string, phoneNumber#:integer]}

Student

matrNumber#	name	Semester#
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	8
...	...	...

- **State**: current state of the data sets
- **Key**: minimal set of attributes, which identify the tuple unambiguously
- **Primary Key**: is underlined
  - One of the key candidates is selected as the primary key
  - Special meaning when referencing tuples

Exercise 1

- Create a Relational model for the course, assistant, and professor classes in the university model. Make sure to identify and underline the primary key.
- The employee class is integrated into the subclasses. You do not need to create a table for them.
- e. g. **Student**: {[matrNo: int, name: String, semester: int]}

Solution Exercise 1

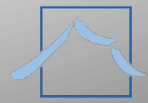
Course: {[courseNumber: int, *titel*: *string, sws*: *int*]}

Professor: {[personalNumber: int, *name*: *string, rank*: *string]*}

Assistant: {[personalNumber: int, *name*: *string, subject*: *string*]}

Primary and Foreign Keys

- Any set of attributes that could be used to uniquely identify any data record in a relational table is called a **candidate key**
- The actual candidate key that is used in the application to identify the records is called the **primary key**
  - The primary key of a table is a set of attributes whose values uniquely identify the data records in the table
- A **foreign key** is an attribute (or a set of attributes) that references the primary key of another table.



Example for Primary and Foreign Keys

User table

Primary key

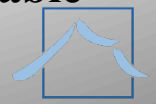
firstName	login	email
“alice”	“am384”	“am384@mail.org”
“john”	“js289”	“john@mail.de”
“bob”	“bd”	“bobd@mail.ch”

Candidate key Candidate key

League table

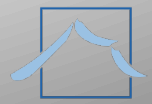
name	login
“tictactoeNovice”	“am384”
“tictactoeExpert”	“bd”
“chessNovice”	“js289”

Foreign key referencing **User table**



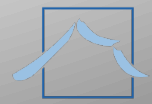
Mapping Associations to Tables in Relational Databases

1. Buried Association
2. Many-To-Many Associations
3. Inheritance
   - Horizontal mapping
   - Vertical mapping.

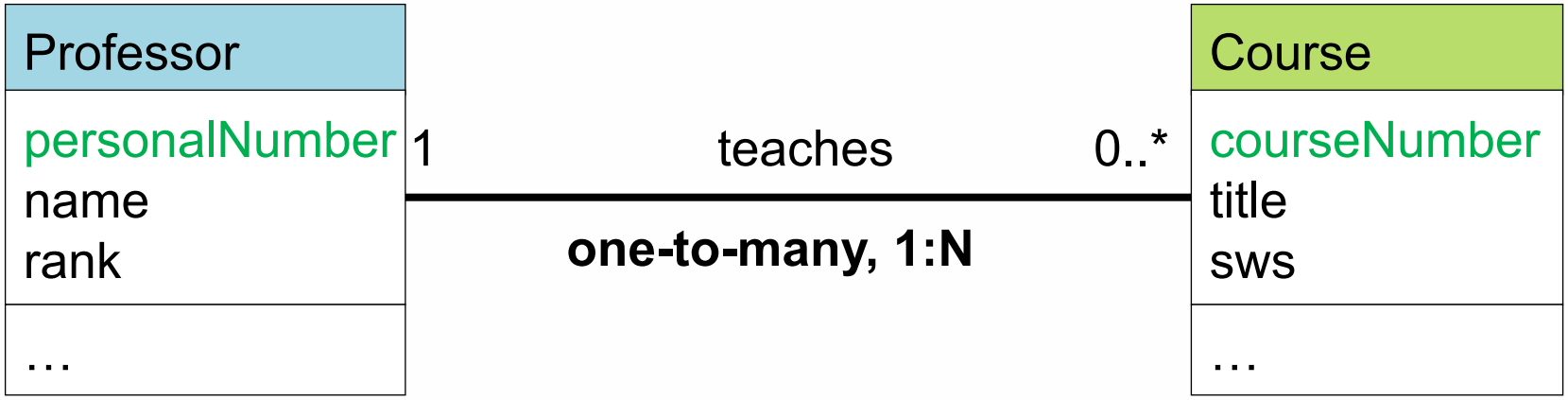


Mapping an Object Model to a Database

- ✓ UML object models can be mapped to relational databases:
  - ✓ Some degradation occurs because all UML constructs must be mapped to a single relational database construct - the **table**
- ✓ Mapping of classes, attributes and associations
  - ✓ Each *class* is mapped to a **Table**
  - ✓ Each class *attribute* is mapped onto a column in the table
    - ✓ The collection of all attributes is called the **Schema**
  - ⇒ ✓ An *instance* of a class represents a row in the table (**Tuple**)
  - – A *one-to-many association* is implemented as buried foreign key
  - – *A many-to-many association* is mapped into its own table
- • Methods are not mapped.



One-To-Many Association – New Table



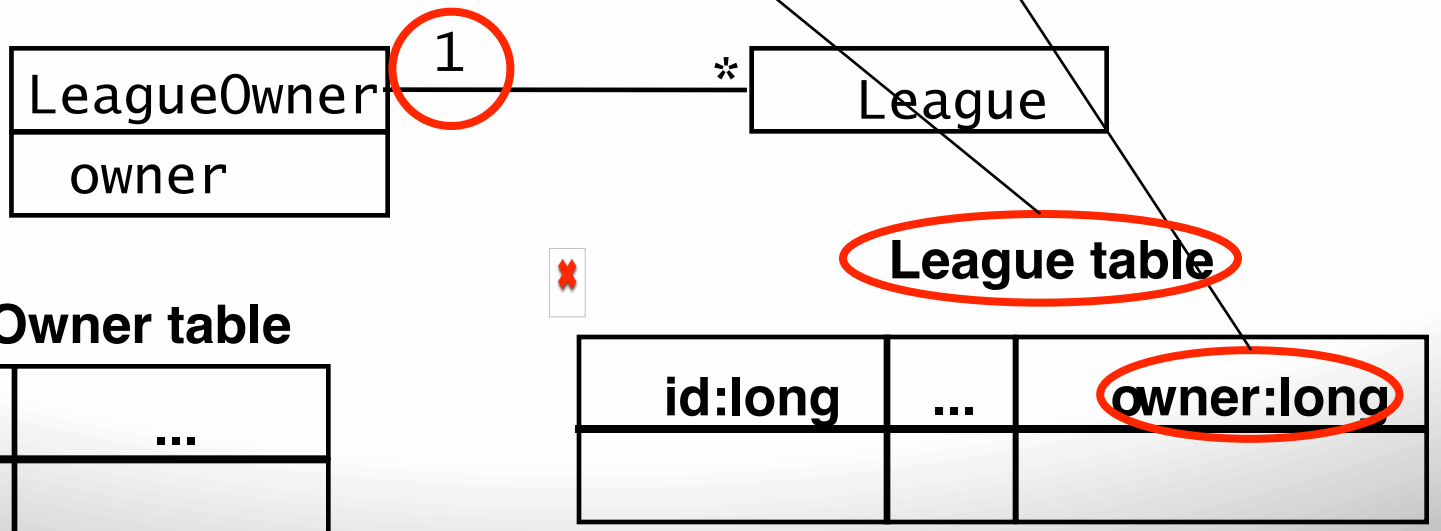
teaches : {[personalNumber: int, courseNumber: int]}

One-To-Many Association - Buried Association

- Associations with multiplicity “one” can be implemented using a foreign key

For one-to-many associations we add the foreign key to the table representing the class on the “many” end

For all other associations we can select either class at the end of the association.



Another Example for Buried Association

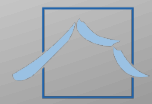
Transaction Table

transactionID	portfolioID

Foreign Key

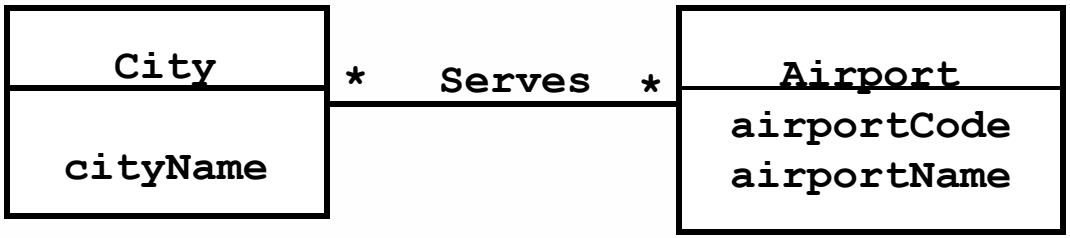
Portfolio Table

portfolioID	...



Mapping Many-To-Many Associations

In this case we need a separate table for the association



Separate table for the association "Serves"

Primary Key

City Table

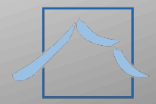
cityName
Houston
Albany
Munich
Hamburg

Airport Table

airportCode	airportName
IAH	Intercontinental
HOU	Hobby
ALB	Albany County
MUC	Munich Airport
HAM	Hamburg Airport

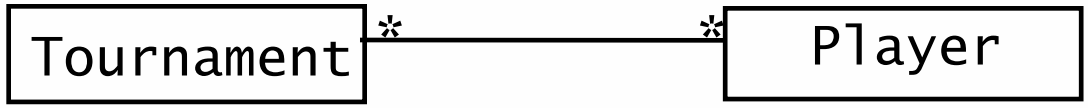
Serves Table

cityName	**airportCode**
Houston	IAH
Houston	HOU
Albany	ALB
Munich	MUC
Hamburg	HAM



Another Many-to-Many Association Mapping

We need the Tournament/Player association as a separate table



Tournament table

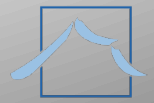
id	name	...
23	novice	
24	expert	

TournamentPlayerAssociation table

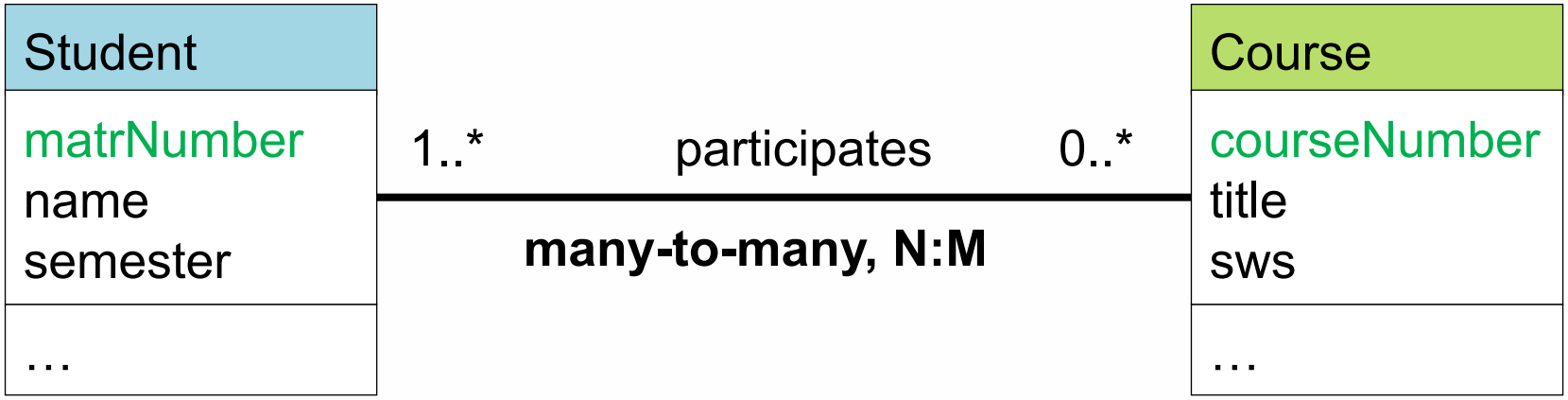
tournament	player
23	56
23	79

Player table

id	name	...
56	alice	
79	john	



Mapping Associations to Relational Model



participates : {[matriculationNr: int, courseNr: int]}

Example state of the association *participates*

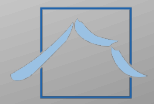
Student	
matrNo	...
26120	...
27550	...
...	...

participates	
matrNo	courseNo
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049

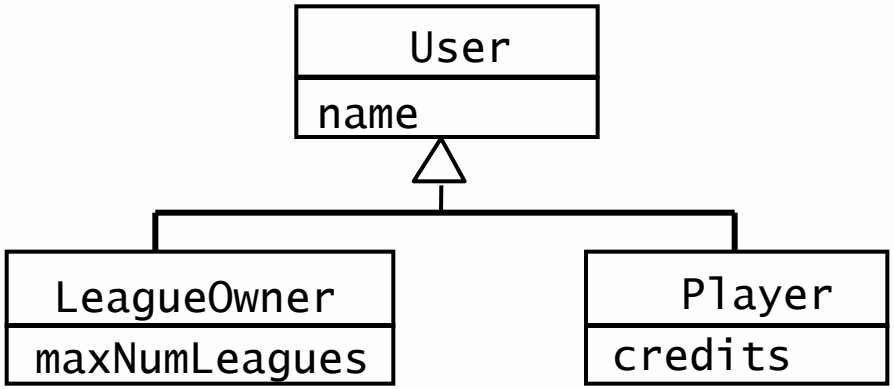
Course	
courseNo	...
5001	...
4052	...
...	...

Realizing Inheritance

- Relational databases do not support inheritance
- Two possibilities to map an inheritance association to a database schema
  - With a separate table (”vertical mapping”)
    - The attributes of the superclass and the subclasses are mapped to different tables
  - By duplicating columns (”horizontal mapping”)
    - There is no table for the superclass
    - Each subclass is mapped to a table containing the attributes of the subclass and the attributes of the superclass



Realizing inheritance with a separate table (Vertical mapping)



User table

id	name	...	role
56	zoe		LeagueOwner
79	john		Player

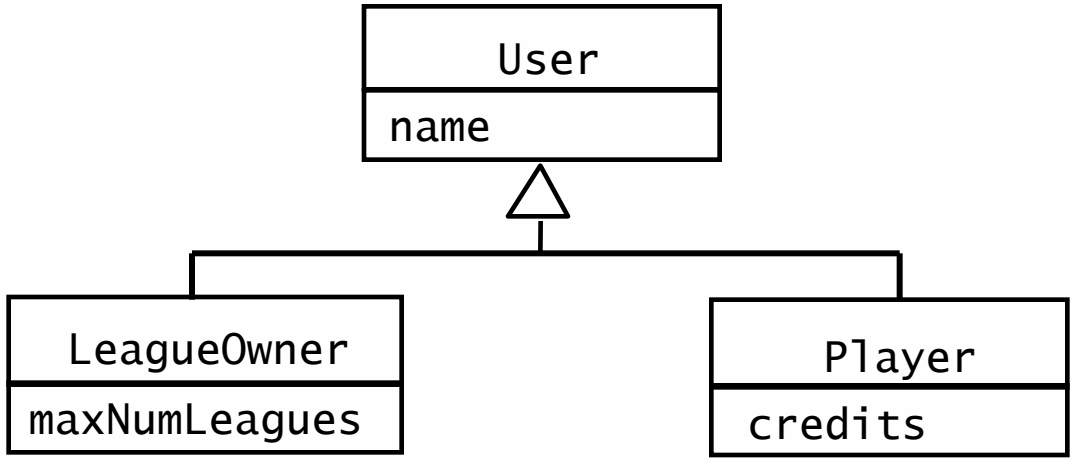
LeagueOwner table

id	maxNumLeagues	...
56	12	

Player table

id	credits	...
79	126	

Realizing inheritance by duplicating columns (Horizontal Mapping)

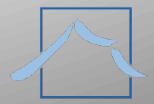


LeagueOwner table

id	name	maxNumLeagues	...
56	zoe	12	

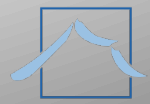
Player table

id	name	credits	...
79	john	126	



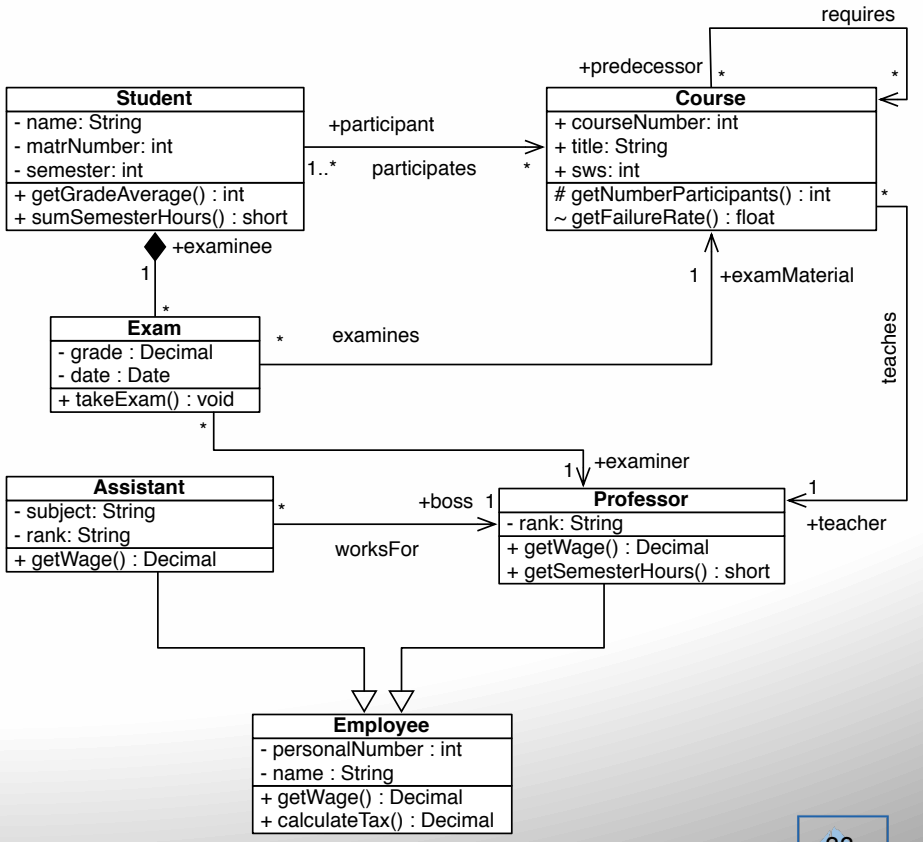
Comparison: Separate Tables vs Duplicated Columns

- The trade-off is between modifiability and response time
  - How likely is a change of the superclass?
  - What are the performance requirements for queries?
- Separate table mapping (Vertical mapping)
  - ☺ We can add attributes to the superclass easily by adding a column to the superclass table
  - ☹ Searching for the attributes of an object requires a join operation.
- Duplicated columns (Horizontal Mapping)
  - ☹ Modifying the database schema is more complex and error-prone
  - ☺ Individual objects are not fragmented across a number of tables, resulting in faster queries



Exercise 2

- Map the associations **worksFor** and **requires** for the university model to a relational model
  e. g. **teaches** : {[personalNo: int, courseNo: int]}
- Mark the primary key
- Which association can you model as a burried association



Exercise 2 - Solution

worksFor: {[assistantPersonalNr: int, *profPersonalNr: int*]}

requires: {[predecessor: int, successor: int]}

Relational model of the university

Professor		
persNo	name	rank
2125	Sokrates	C4
2126	Russel	C4
2127	Kopernikus	C3
2133	Popper	C3
2134	Augustinus	C3
2136	Curie	C4
2137	Kant	C4

Student		
matrNo	name	semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Course			
courseNo	title	sws	taughtBy
5041	Ethics	4	2125
5043	Cognitive Science	3	2126
5049	Maieutics	2	2125
4052	Logic	4	2125
5052	Philosophy of Science	3	2126
5259	Vienna Circle	2	2133
5022	Belief and Knowledge	2	2134

requires	
predecessor	successor
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

participates	
matrNo	courseNo
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistant			
persNo	name	subject	boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

examines			
matrNo	courseNo	persNo	grade
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2