Introduction to Relational Databases

- Bachelor CS, Lille 1 University
- Sept. 28th, 2011 (lecture 5/12)
- Today's lecturer: A. Bonifati
- Topic: Introduction to Database Modeling (aka Conceptual Modeling)

Prologue: a Class Diagram in UML

- Today's lecture: where are we?
  - we need to do an analysis of the purpose of a database and the objectives of a database modeling
- Goals of the conceptual model
  - focus on the concepts of the real world
  - by somehow avoiding to think about the implementation of those concepts (as objects in a programming language or relations in a database)

A Class Diagram in UML

- With UML one can stay with an high level of abstraction
  - no details for the classes
  - no methods for the classes
  - no need to have all attributes in the relations
  - no need to have the types of the attributes
- Whereas all the above is relevant for the logical model (i.e. in SQL creation scripts)

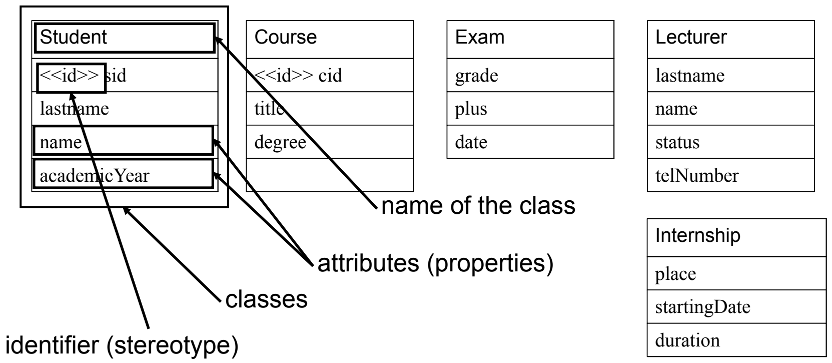
Example: a Database of information about courses

- It is necessary to implement a database used by the administrative staff of a given university to handle courses in Computer Science
- Such a system has to manage the data for Bachelor degrees. For each student, one must register the exams that he/she has passed.
- Such a system has also to include the data about courses and about the exams that are previewed for each course.
- For each course one has to keep track of the lecturer, that can be one or more;
- For each lecturer one has to keep track of the phone numbers;
- For the students, it is necessary to record the professor who is responsible of the final internship and also the information about the internship. Only students enrolled into the third year can have such an advisor for the internship period.

Class

- Abstract “concepts” from real world
  - facts, people, things, that exist autonomously
  - example: student, course, exam, lecturer
- Instance of a class
  - set of objects representing the reality
- Classes have attributes
  - attributes are properties of a class that are relevant for the application
  - stereotype <<id>> for identifiers (UML syntax)
  - stereotype: UML notation for constructs that have a precise role

Classes: 5 examples



Student
<<id>> sid
lastname
name
academicYear

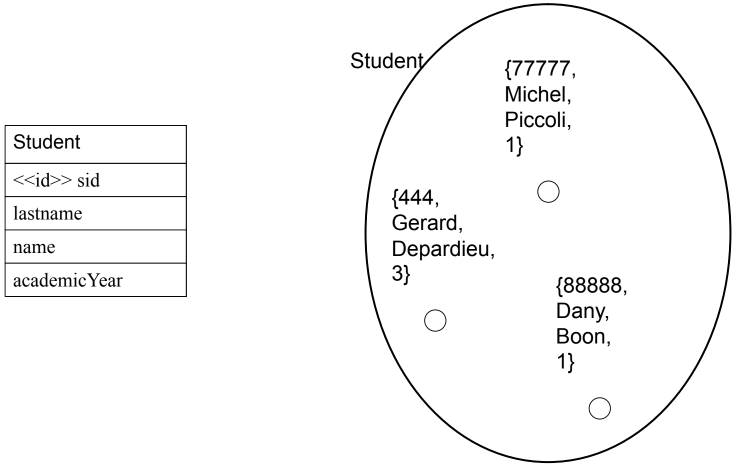
Course
<<id>> cid
title
degree

Exam
grade
plus
date

Lecturer
lastname
name
status
telNumber

Internship
place
startingDate
duration

Class student: instance

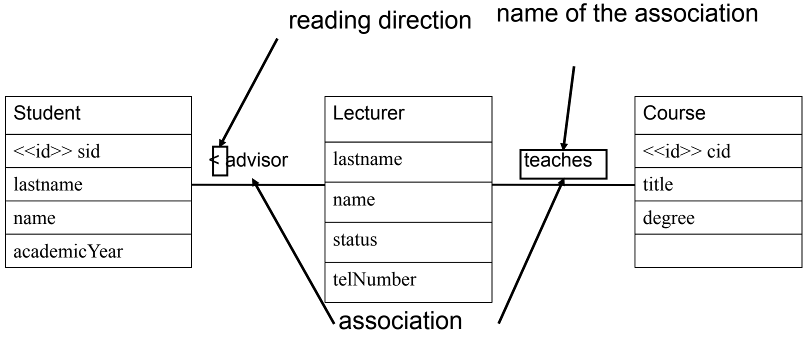


Student
<<id>> sid
lastname
name
academicYear

Associations

- Relations between classes
  - logical relationship relevant for the application
  - ex: a student has passed an exam
  - ex: a lecturer teaches a course
- Instance of an association
  - set of arrows among instances of the involved classes

Associations: examples



reading direction

name of the association

Student

<<id>> sid

lastname

name

academicYear

< advisor

Lecturer

lastname

name

status

telNumber

teaches

Course

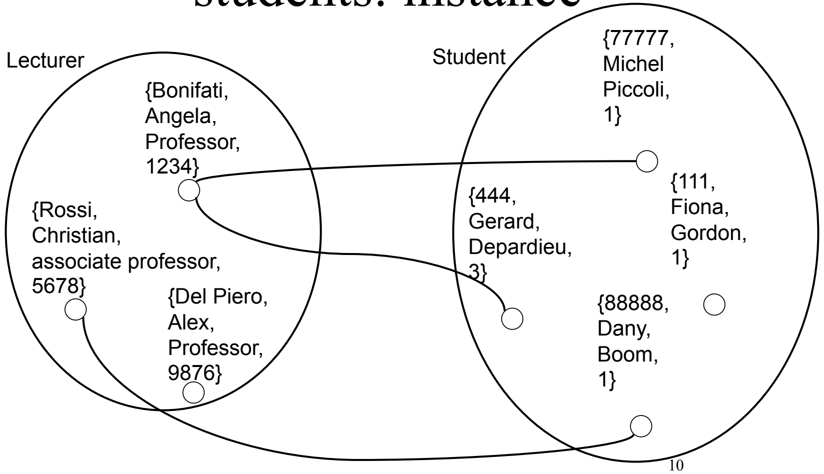
<<id>> cid

title

degree

association

Association between lecturer and students: instance



Lecturer

{Bonifati, Angela, Professor, 1234}

{Rossi, Christian, associate professor, 5678}

{Del Piero, Alex, Professor, 9876}

Student

{77777, Michel Piccoli, 1}

{444, Gerard, Depardieu, 3}

{111, Fiona, Gordon, 1}

{88888, Dany, Boom, 1}

Cardinalities

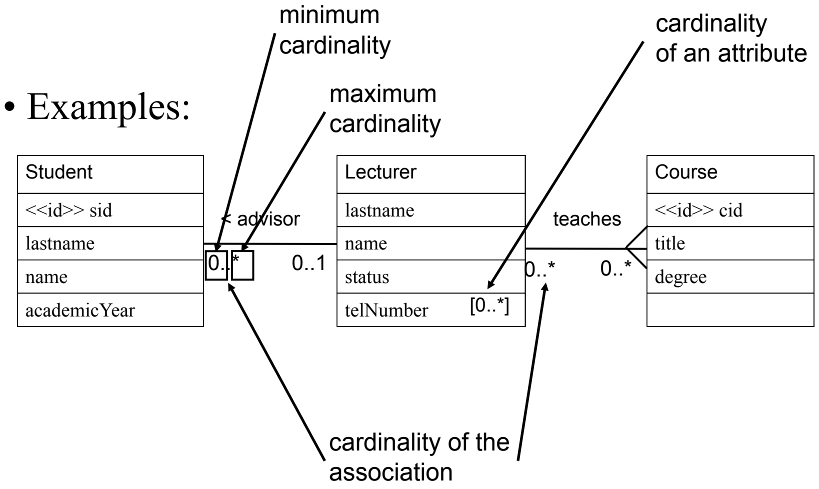
- Constraints on the associations
  - constraints on the number of edges between objects
- Constraints on the attributes
  - how many values for the attributes?
- Minimum cardinality
  - typically 0 or 1
- Maximum cardinality
  - typically 1 or * (n) (but also 3 or 5)

Cardinality

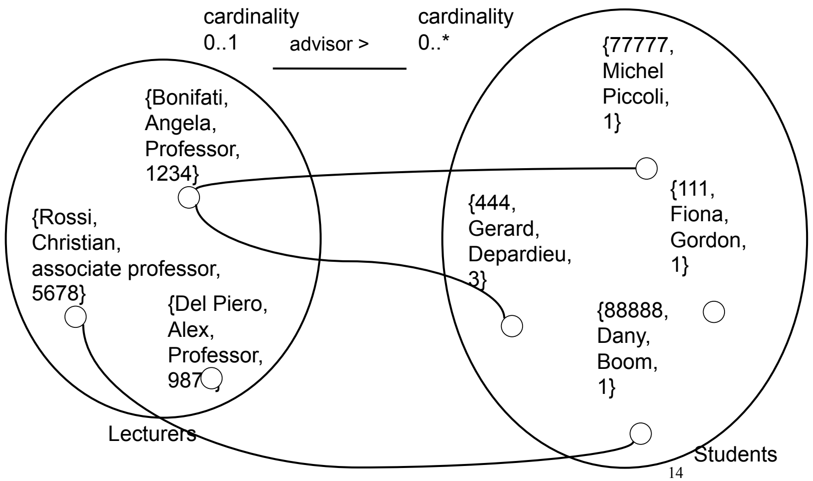
- Cardinality of an association
  - it has to be expressed for both classes connected by an association
  - four cardinalities (a pair min-max for each direction; some can be omitted though)
- Cardinality of a class within an association
  - Min/Max number of objects of a class that can be associated to the objects of the other class

Cardinality

- Examples:



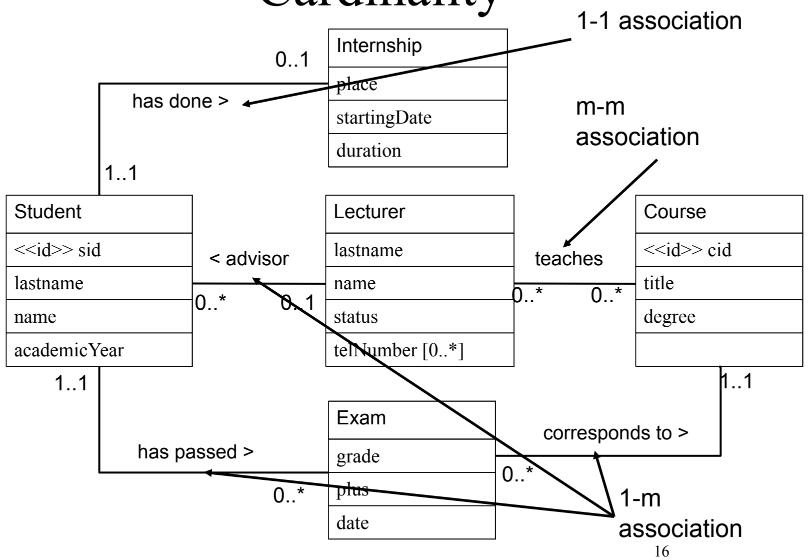
Cardinality



Cardinality

- Different kinds of cardinalities for associations
  - 1 to 1: max cardinality equal to 1 on both sides
  - 1 to many: max cardinality equal to 1 on one side and * on the other side
  - many to many: max cardinality equal to * on both sides

Cardinality

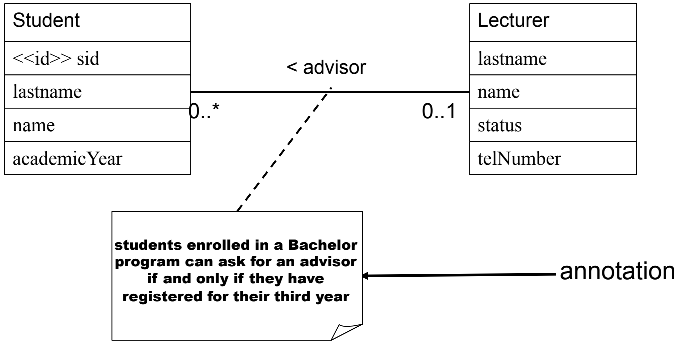


The complete diagram

- The complete diagram
  - contains all the revised constructs
- It is possible to add annotations
  - useful to add comments
  - and to express in UML constraints that are otherwise not expressible, such as: the students enrolled in a Bachelor program can ask for an advisor if and only if they have registered for their third year!

Towards the complete diagram

- Examples:



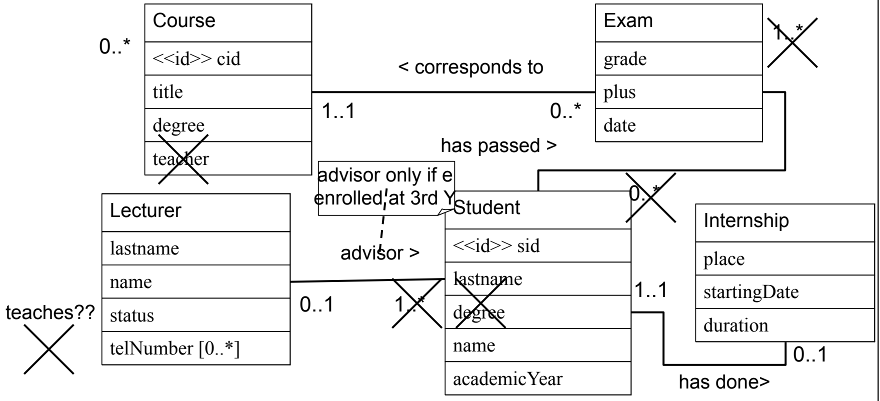
Student: <<id>> sid, lastname, name, academicYear

Lecturer: lastname, name, status, telNumber

< advisor (Student 0..* — Lecturer 0..1)

students enrolled in a Bachelor program can ask for an advisor if and only if they have registered for their third year

annotation



Course: <<id>> cid, title, degree, teacher

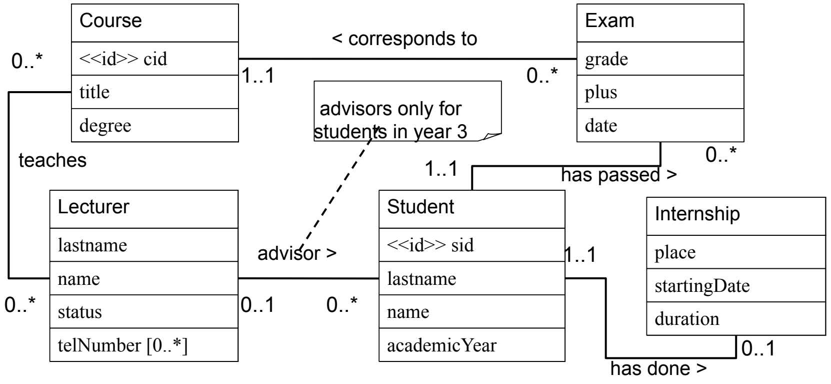
Exam: grade, plus, date

Lecturer: lastname, name, status, telNumber [0..*]

Student: <<id>> sid, lastname, degree, name, academicYear

Internship: place, startingDate, duration

A diagram with mistakes (done on purpose :-)



Course: <<id>> cid, title, degree

Exam: grade, plus, date

Lecturer: lastname, name, status, telNumber [0..*]

Student: <<id>> sid, lastname, name, academicYear

Internship: place, startingDate, duration

The Corrected Diagram

Identifiers

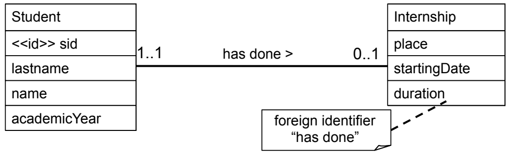
- Within classes
  - not all classes have explicit identifiers
  - but the objects of a class have to be distinct (ex: two apples within the class “Apples”)
  - in a sw program, all objects have an implicit identifier (hidden): OID
  - explicit identifiers (e.g. sid, ssn, plate nr.) exist for a catalog
  - they may become relevant inside a database model

Classes

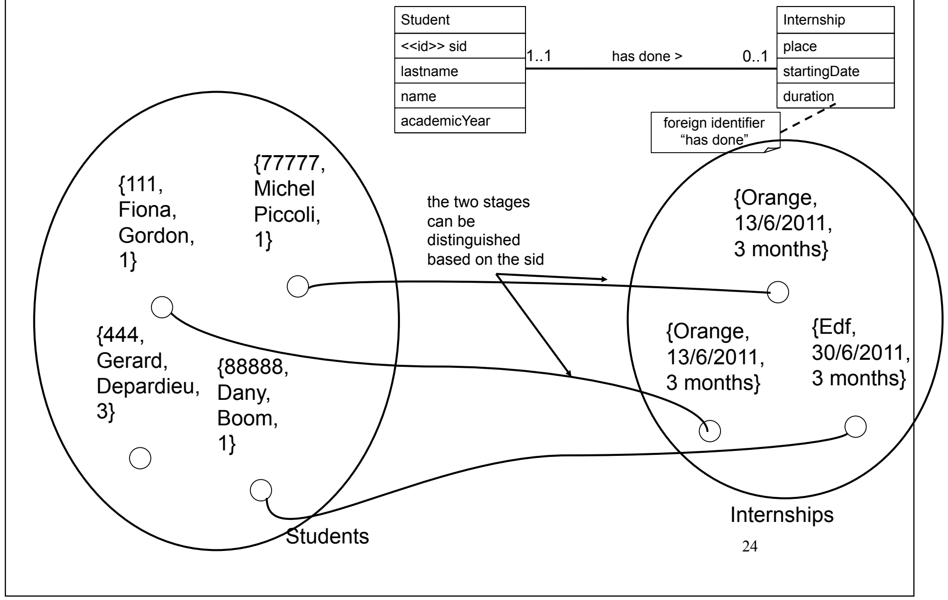
- Foreign Identifiers
  - some objects are identified thanks to other objects to which they are associated
  - example: a Internship is identified by means of the student ID (the student who has done it)
  - to identify a Internship it is necessary to traverse the edge that represents the association
  - the cardinality on the other side has to be 1

Classes

- Foreign identifier
  - it is represented as an annotation
- Example: Internship



Classes



Cardinalities

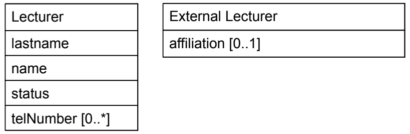
- Cardinalities
  - constraints on the instances of attributes and associations
- Cardinality of attributes
  - constraints on the number of values of an attribute
- Cardinality of an association
  - constraints on the number of edges of an association

Cardinalities

- Minimum cardinality
  - typically 0 or 1
  - 1 is seldom used
- Maximum cardinality
  - typically 1 or *
- There may be other numbers
  - e.g.: association between Match and Team 2..2
- Short notation
  - 1 stands for 1..1; * stands for 0..*

Cardinality

- For the attributes
  - standard cardinality1..1
  - minimum cardinality 0: optional attribute
  - maximum cardinality *: multivalued attribute



Lecturer
lastname
name
status
telNumber [0..*]

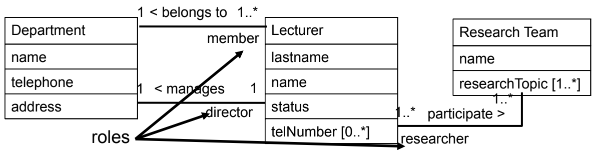
External Lecturer
affiliation [0..1]

Association

- Logical relationship between classes
- Instance of an association
  - edge that connects objects of the two involved classes
- Cardinality
  - constraint on the number of edges for an object
- Different kinds of cardinalities
  - One to one: both maximal cardinalities are 1
  - One to many: maximal cardinalities 1 and *
  - Many to many: maximal cardinalities * and *

Association

- Role:
  - description of the role of a class in the relationship



Department: name, telephone, address

Lecturer: lastname, name, status, telNumber [0..*]

Research Team: name, researchTopic [1..*]

1 < belongs to 1..* member

1 < manages 1 director

1..* participate > 1..* researcher

roles

Association

- Typically
  - binary associations (between 2 distinct classes)
- But there may exist
  - recursive associations: associations between objects of the same class
  - n-ary associations (seldom used): associations that involve objects of multiple classes (3 or more)

Summary of today's lecture

- Introduction
  - Requirement Analysis
- Class diagram in UML
  - Class
  - Association
  - Cardinality
- The complete diagram
- Foreign identifiers, cardinalities of associations and attributes

Postgres commands (related to TP on 26/09/2011)

- Warning: Export in pgAdmin does not work (not clear why -- it might be a problem with the installation on the machines in the Lab); in order to see the commands and thus fill your TP exercise, type inside a console:
  - pg_dump -h webtp.fil.univ-lille1.fr - U bonifati -W bonifat > db.out (Il faut remplacer bonifati avec son propre nome d'utilisateur).
  - (Important) Notice that you only need to copy the commands that you need from db.out; please avoid delivering the entire file that contains the entire database and thus a lot of other stuff.
- Tips:
  - once you have prepared your scripts (e.g. creationDBbonifati.sql, insertionDBbonifati.sql), you can double-check that they work properly by reloading them into the database:
    - by using Import in pgAdmin (or by using psql in the console -- in such a case, enclose your creation commands into begin transaction; ...commit; and do the same for the modification commands)