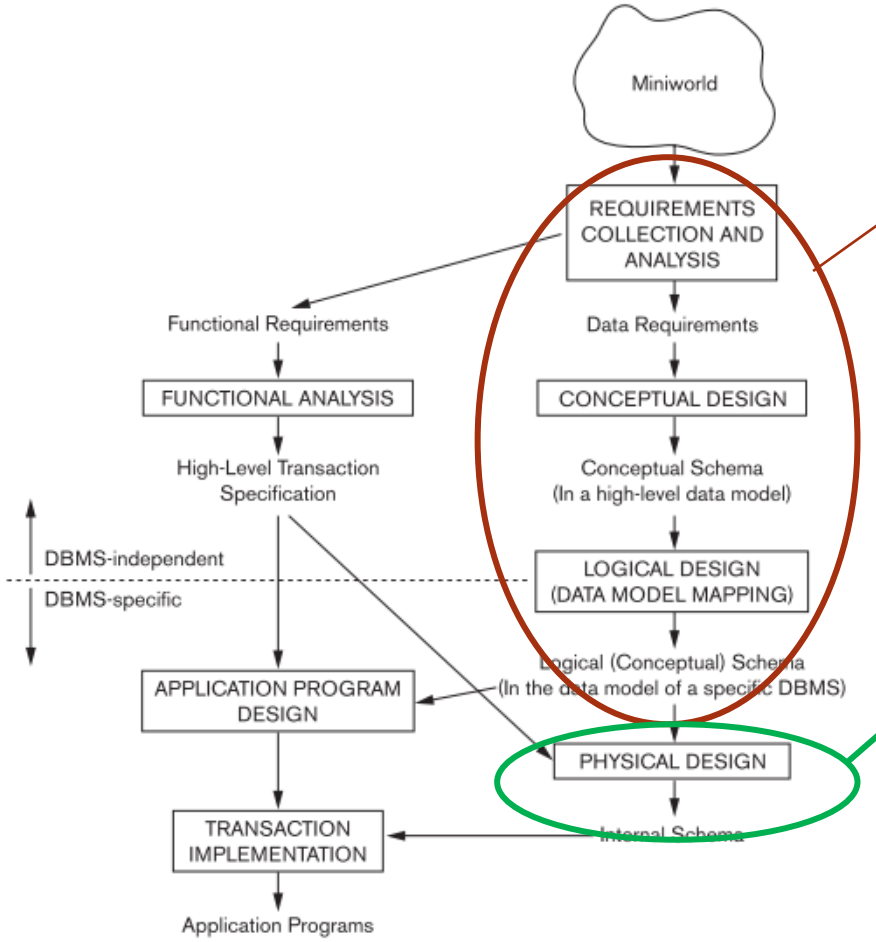
SQL Workshop

Data Modelling Using UML

sjapit@uow.edu.au

19 March 2024

Database Design Process



For this workshop, we do a quick revision on these topics.

More exercises on this topic.

Overview

Object:

- An object is simply something that makes sense in an application context.
  - It can be a tangible entity such as person, or
  - it can be a less tangible element such as an event.

  E.g., a student may be an object, and an examination might also be an object.

Overview

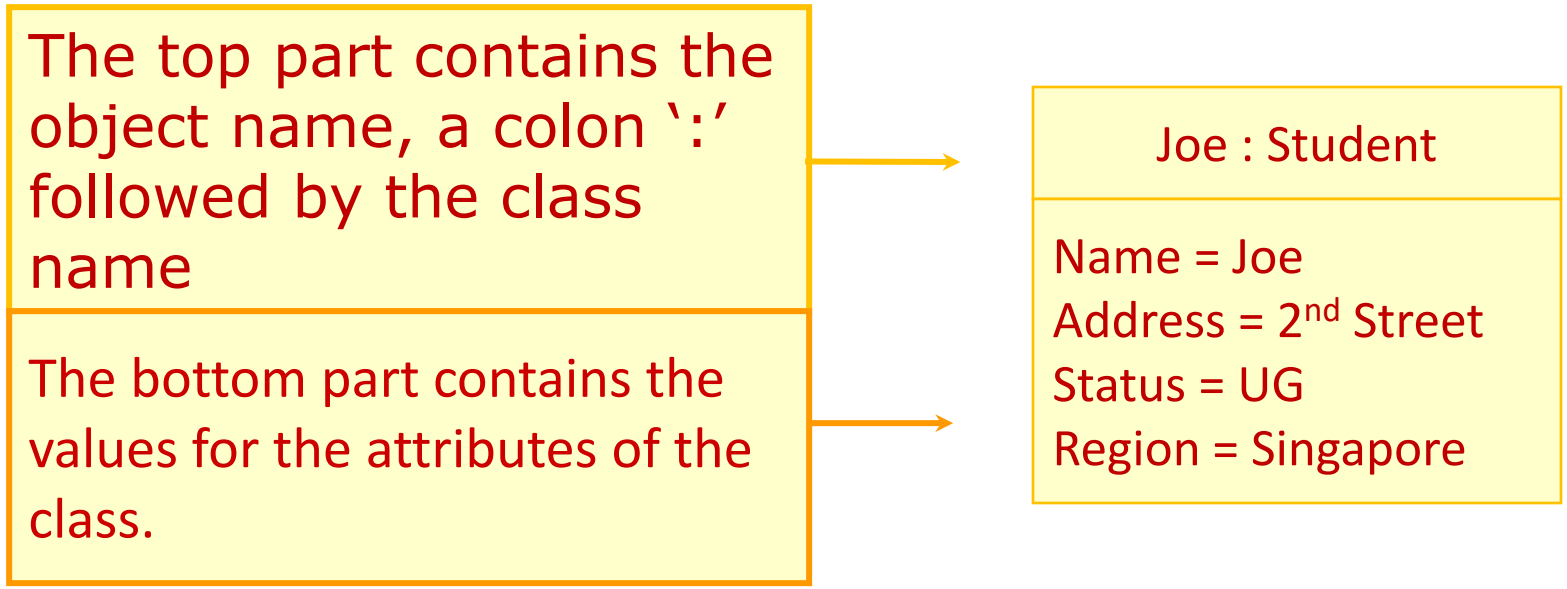
An object has the following aspects as its component:

1. A Collection of data (attributes or properties) which defines the **characteristics** of the object,
2. The functional logic (methods) which defines the **behaviour** of the object, and
3. Its identity – an **identification** which serves to distinguish it from all others.

Overview

Modelling an object:

- An object is drawn as a box with two parts:



Overview

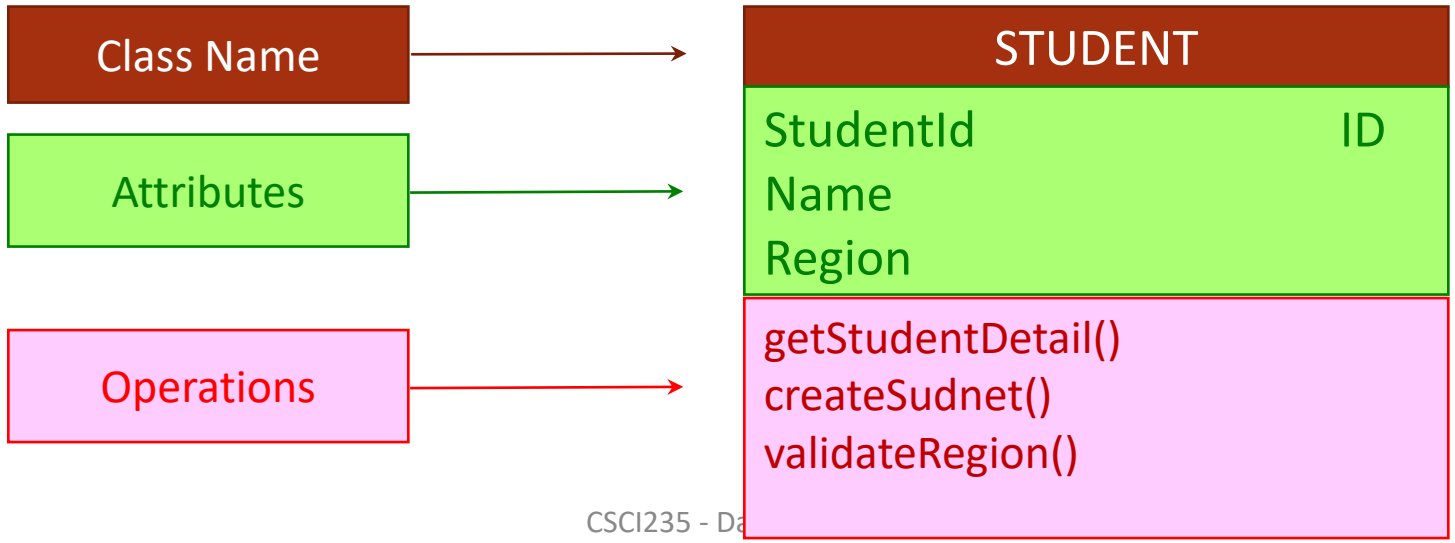
Class:

- Classes *define* the types of **objects** that exist within the system. It represents entities with common characteristics and behavior; i.e., attributes, services, rules or constraints, and relationships.
- A class has both an interface and a structure. The interface describes how a class and its instances can be interacted with via methods, while the structure describes how the data is partitioned into attributes within an instance.

Overview

Modeling a class:

- A class is drawn as a three-part box, with the class name in the top part, a list of attributes (with optional types) in the middle part, and a list of operations (with optional argument lists and return types) in the bottom part.



Overview

- Two types of class:
  - A **catalogue class** is a class whose objects play a role of templates for the objects from the respective **physical class**.
    - A catalogue class is a class whose objects are reusable across multiple assemblies.
  - A **physical class** is a class whose objects are dedicated to at most one assembly.
- A catalogue class is a description of a respective physical class.

Overview

Example:

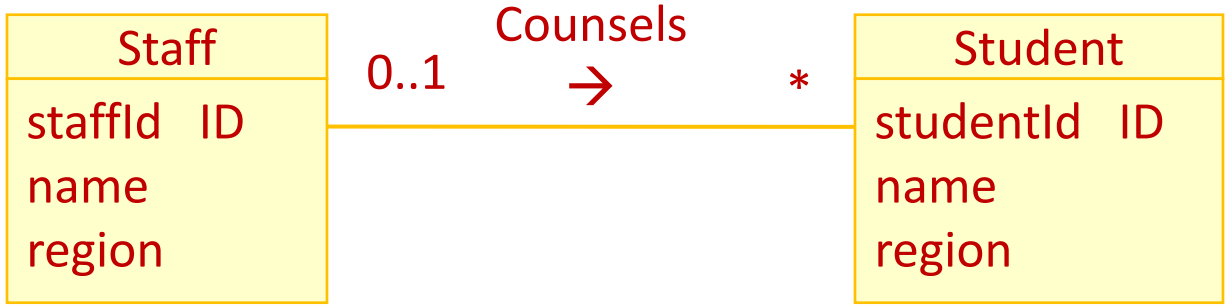
Physical classes	Catalogue classes
Book copy	Book
Room	Room type
Flight instance	Flight
Keyboard	Keyboard blueprint
Running course	Course
Assignment	Assignment specifications

Overview

Links and Associations

- Means for establishing **relationships (conceptual connection)** among objects and classes.
- Associations represent **structural relationships** between objects of different classes.

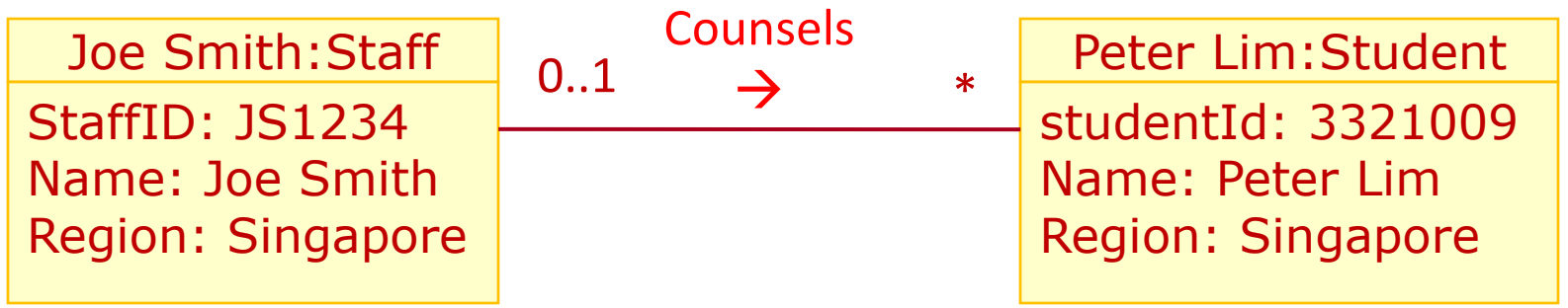
For example: a **staff** *counsels* **students**



Overview

- The individual instances of an association are called links.
- A link is a physical or conceptual connection between object instances.

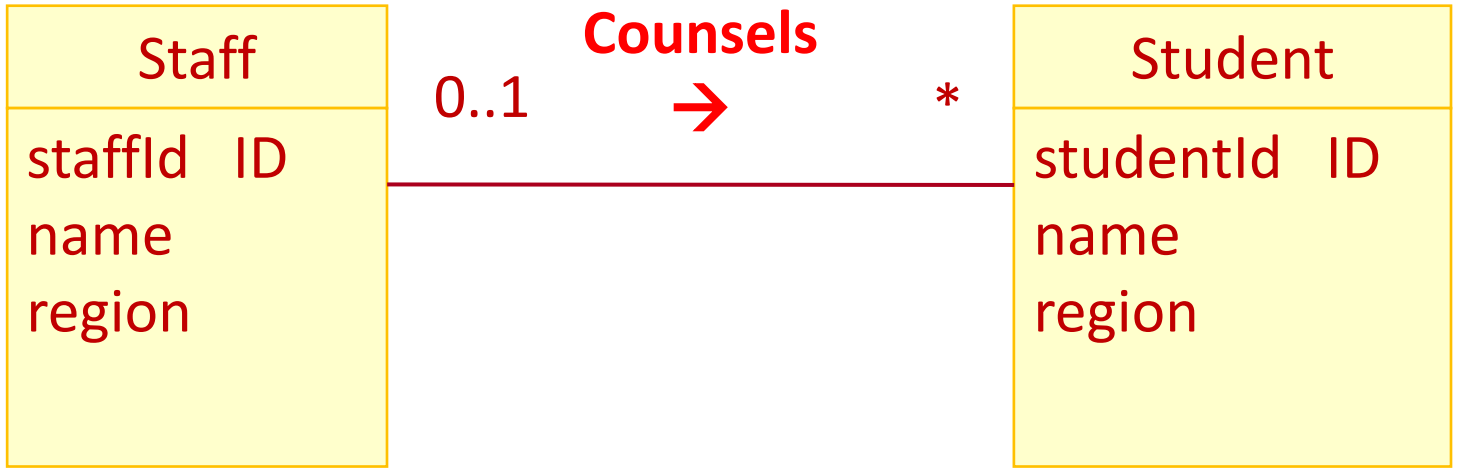
For example: **Joe Smith** *counsels* **Peter Lim**



Overview

Name and direction:

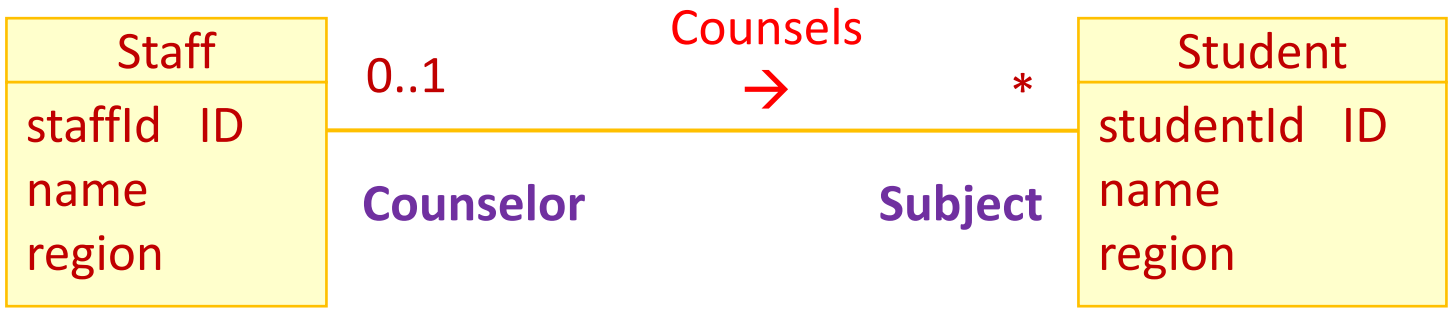
- Most associations are binary, drawn as lines between pairs of classes.
- An association **has a name** with an optional arrow showing which way it is read.



Overview

Role:

- Each end of an association is a **role**, and each role can have a name (role name), showing how its class is viewed by the other class.



- The role names opposite a class must be unique.

Overview

Multiplicity:

- Multiplicity indicates the number of instances of one class that may relate to a single instance of an associated class.
- Examples of multiplicity notation:

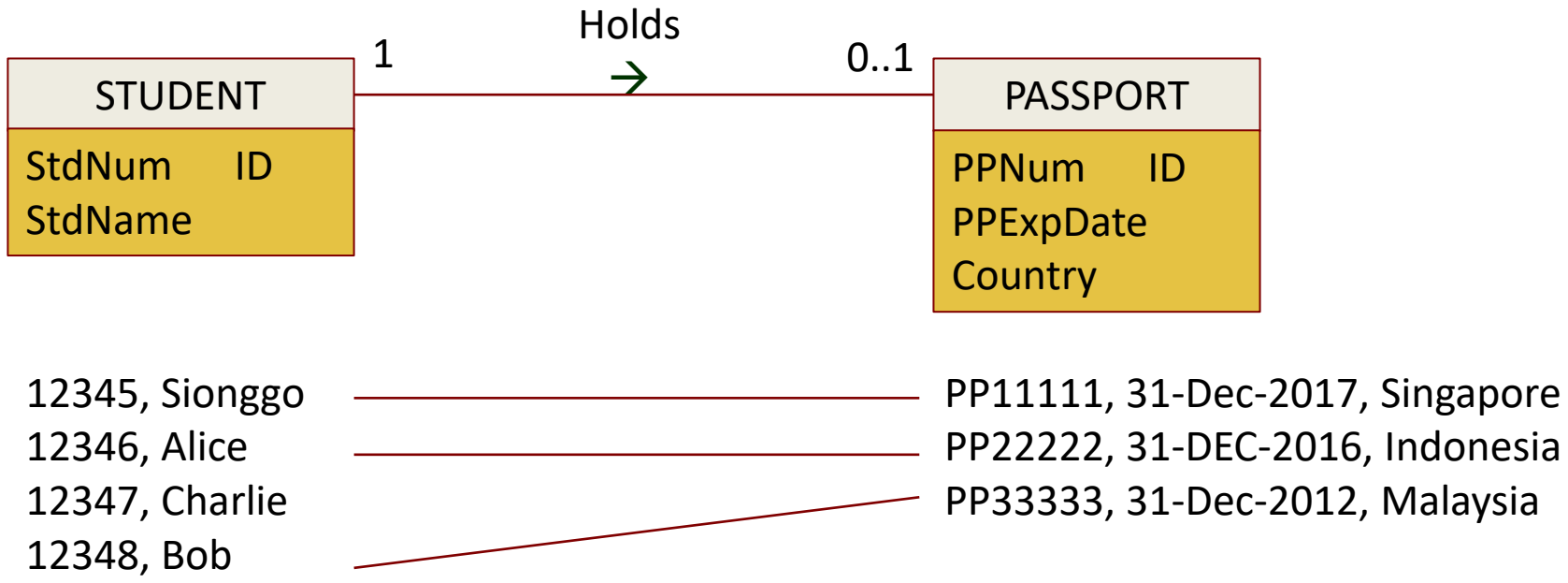
Notation	Meaning
0..1 or none	No instances, or one instance. May mean optional.
0..* or *	Zero or more instances. May mean optional.
1..*	One or more instances (at least one).
2..4	Two to four inclusive.
1	Exactly one instance. (A blank (no notation) means zero or one.)
3	Exactly three instances.

Overview

One-to-one Multiplicity

For examples:

- Each student may have a passport.

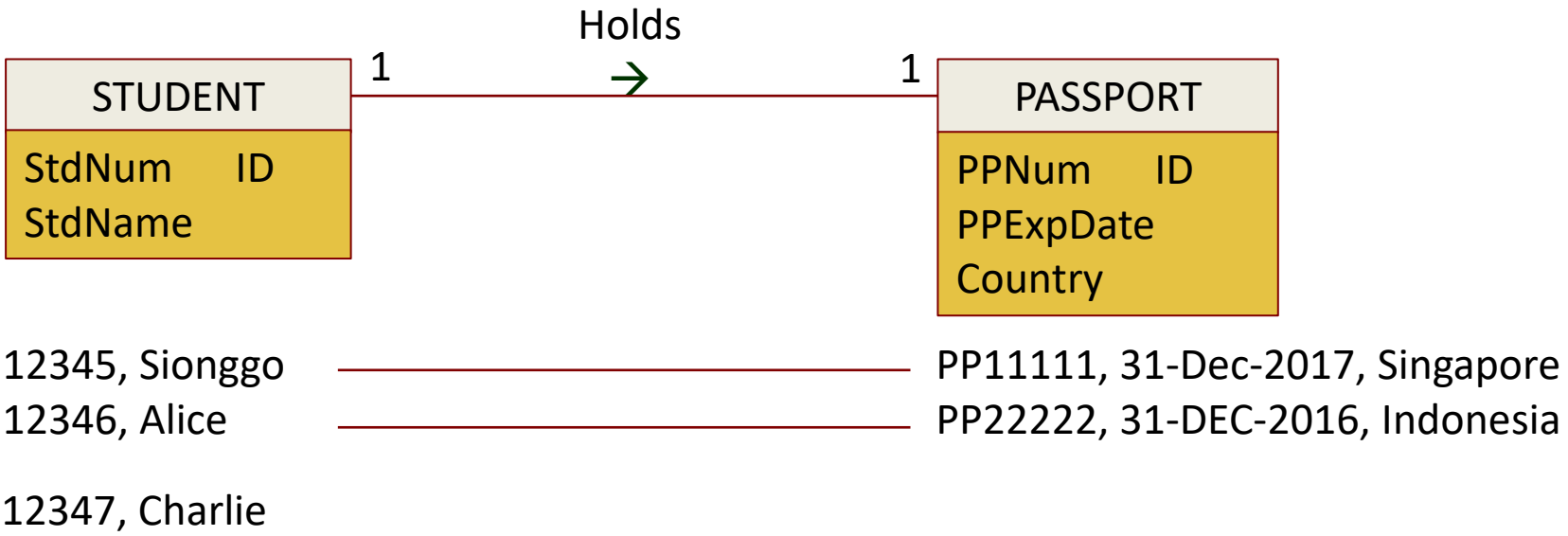


Overview

One-to-one Multiplicity

For examples:

- Each student must have a passport.

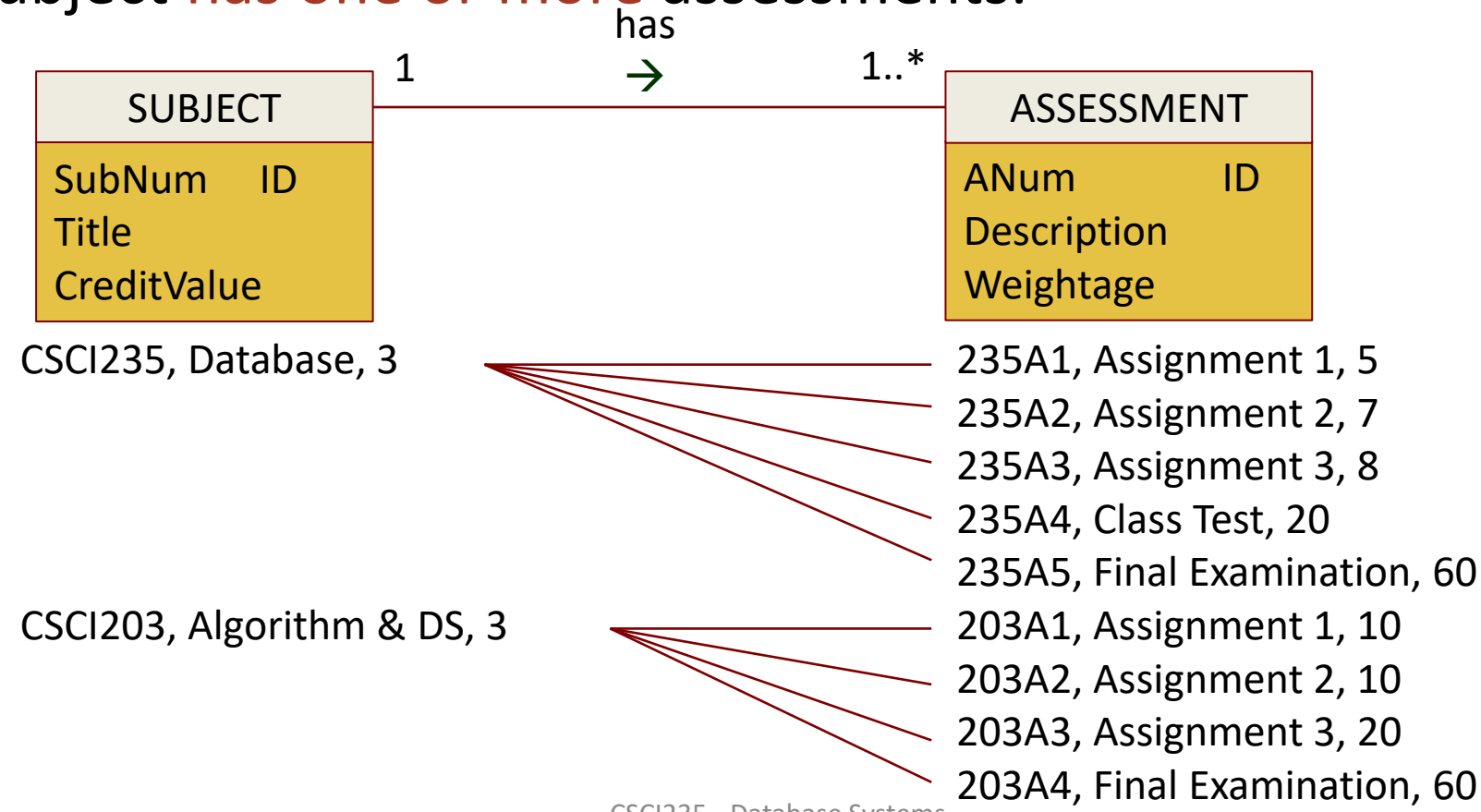


This is not allowed.

Overview

One-to-many multiplicity

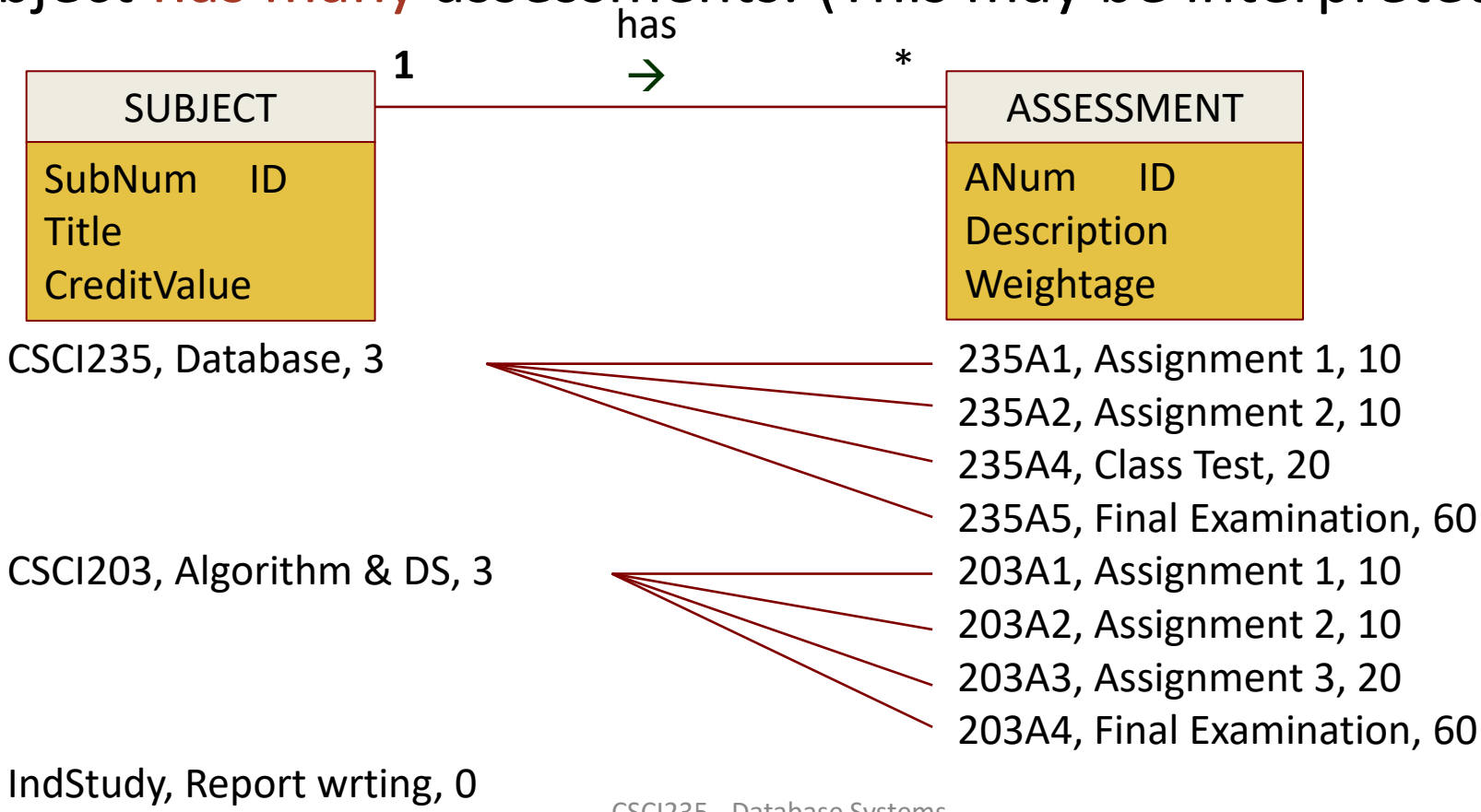
- Each subject has one or more assessments.



Overview

One-to-many multiplicity

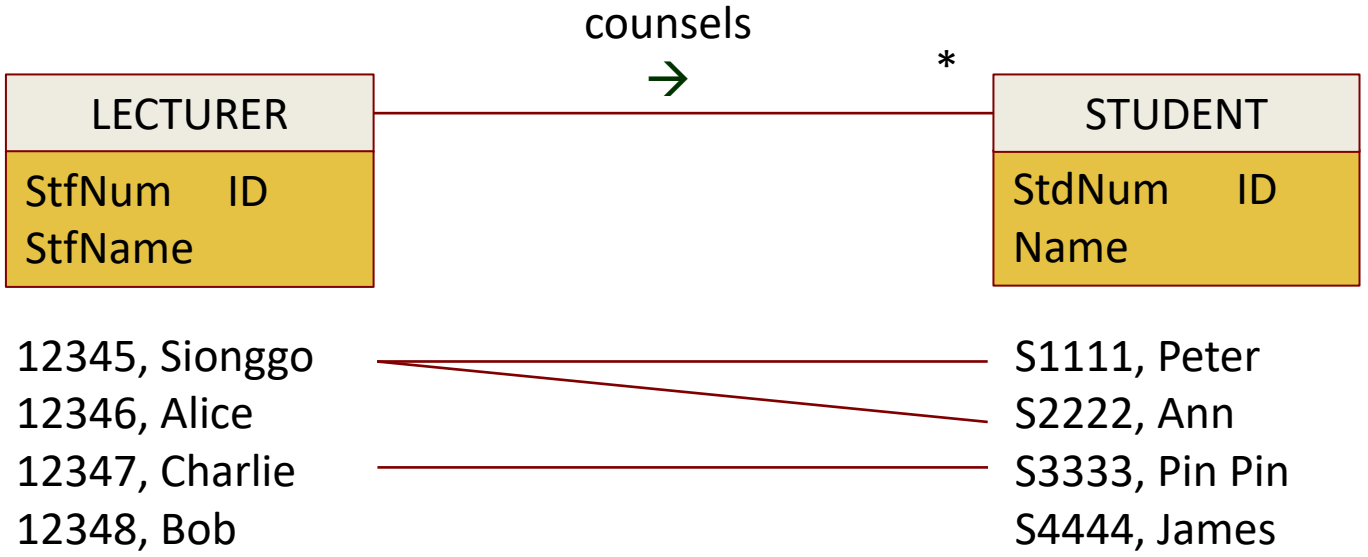
- Each subject has many assessments. (This may be interpreted as optional.)



Overview

One-to-many multiplicity

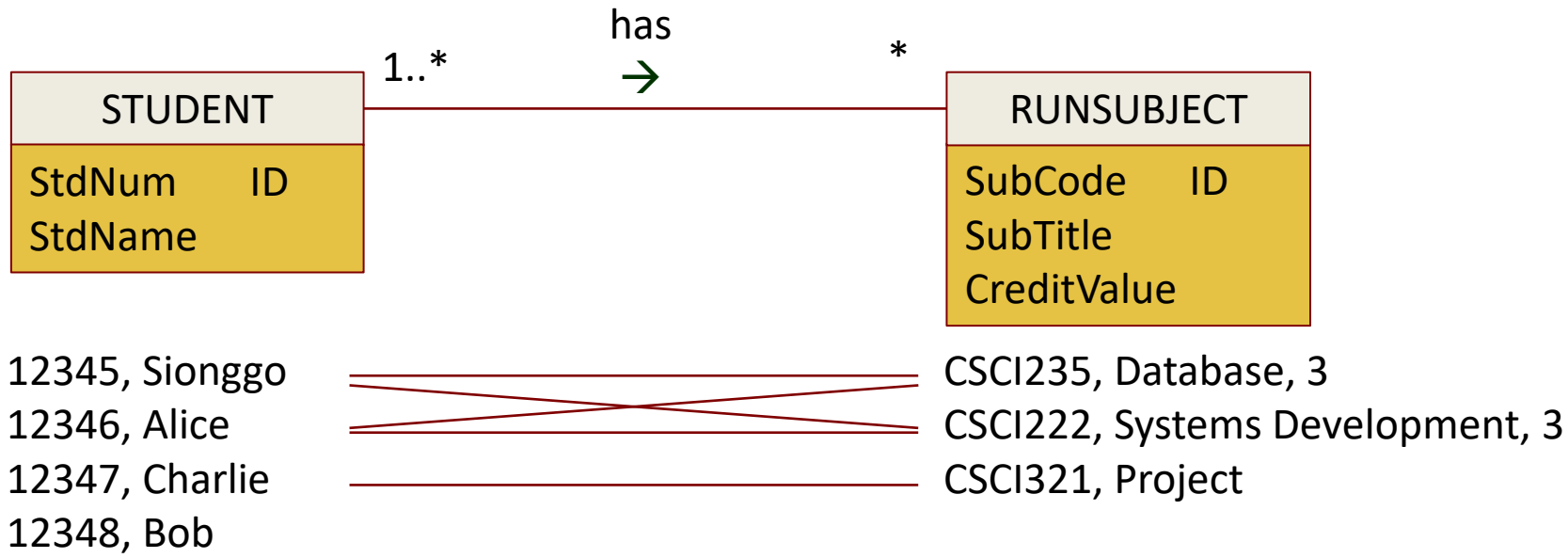
- A lecturer may counsel many students.



Overview

Many-to-many multiplicity

- E.g., Students may register to one or more subjects and each subject has one or more students register to it.



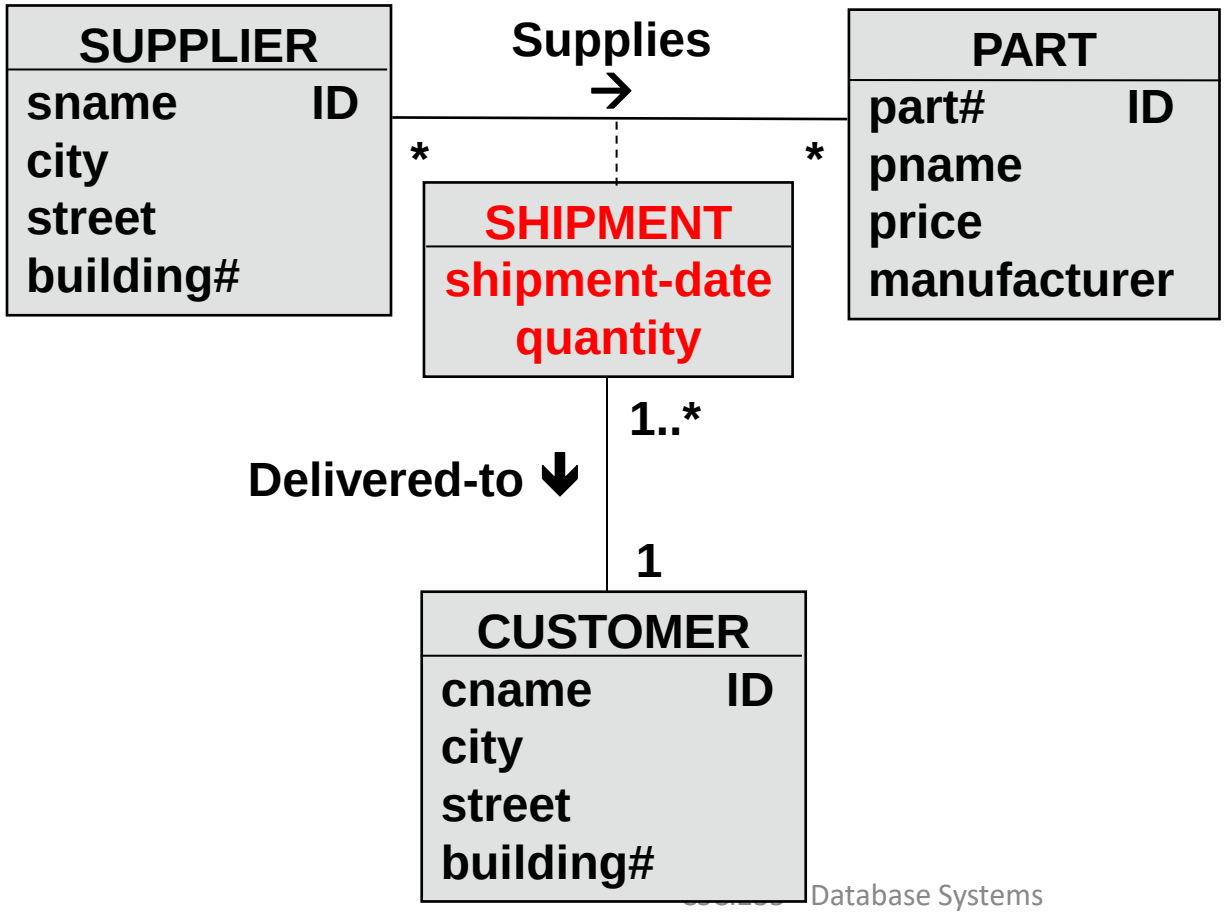
Overview

Association Class

- An association class is a class that is describing the many-to-many association.
- An association class is denoted by a box attached to an association by a dashed line.

Overview

- An example of an association class

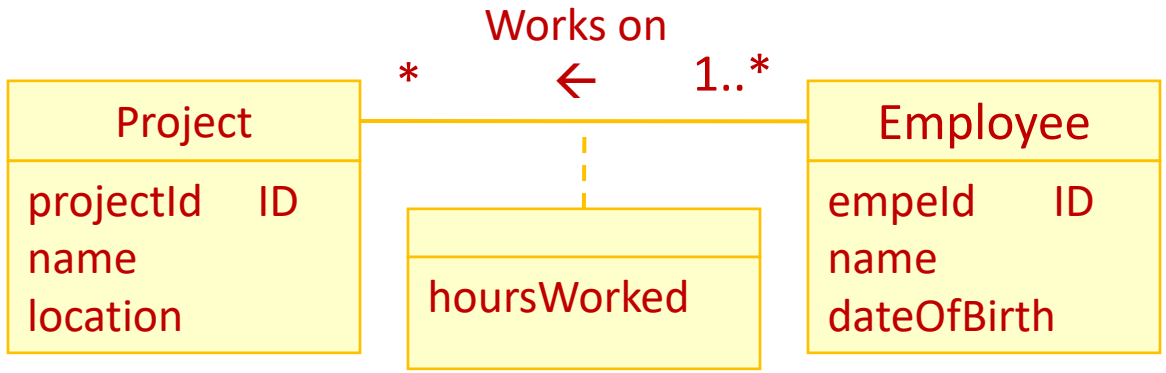


Overview

Link Attribute

- A link attribute is a value held by an association and not belonging to either class by itself.

  For example, In an organization, employees work in multiple projects and a project has multiple employees work on. The respective department want to keep track of the number of hours the employee works on each project.



Overview

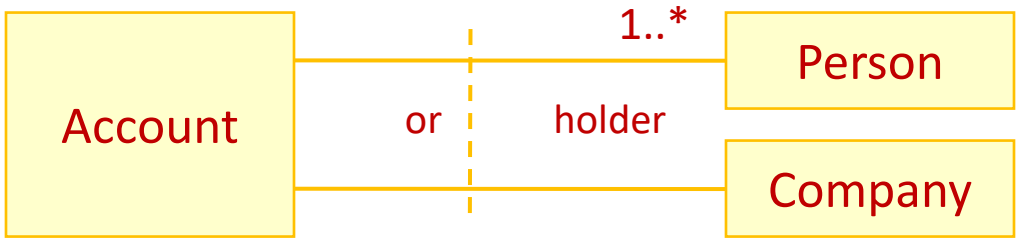
Types of association:

- Exclusive-or association
- Qualified associations
- Specialization/Generalization
- Composition/Aggregation

Overview

- Occasionally one class can participate in two associations, with the restriction that each object can only participate in one of the associations at a time. This can be shown by placing an “or” constraint between the pair of associations.

An example of or-association:



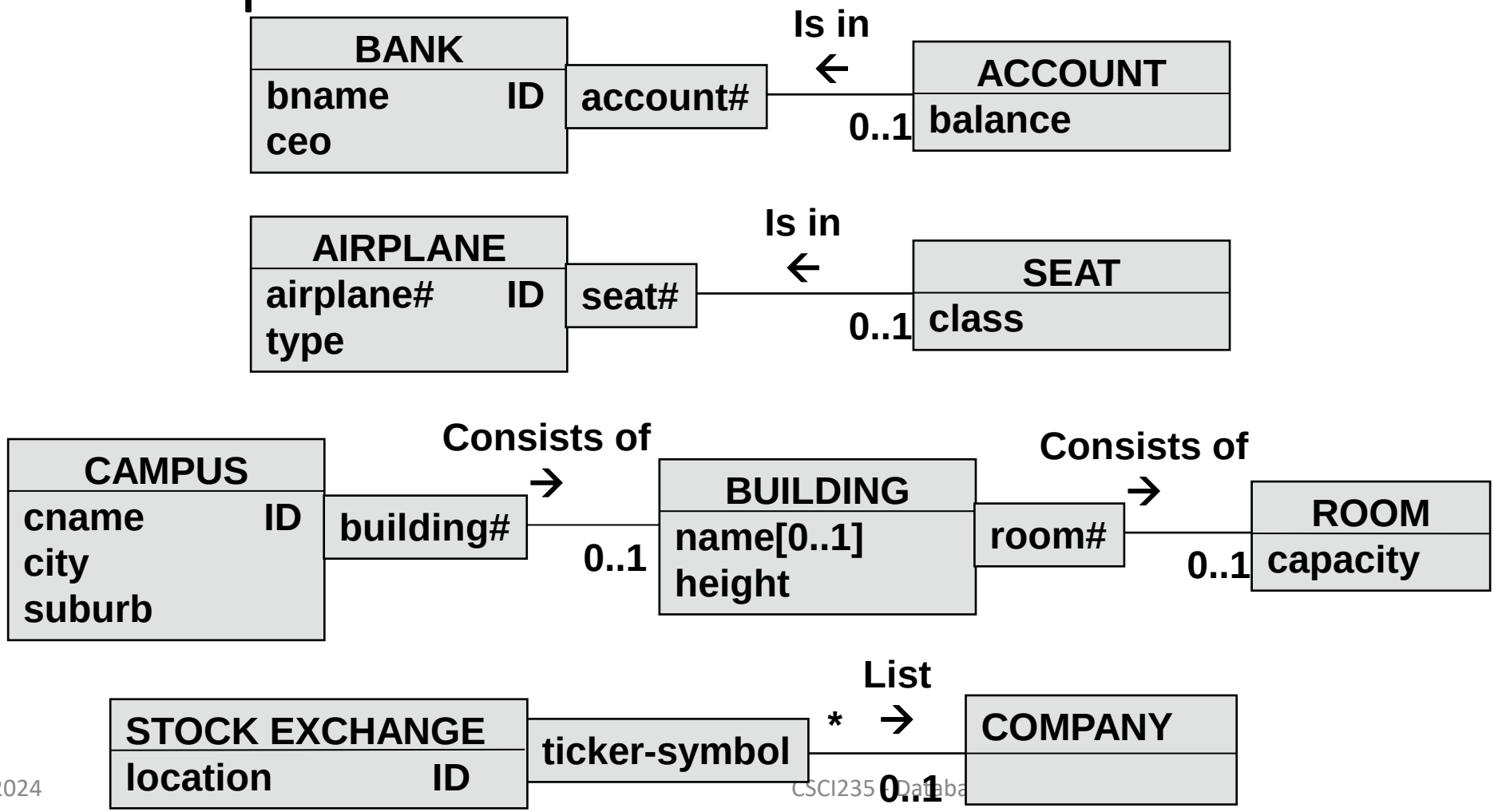
Overview

Qualified Associations:

- A **qualified association** has a **qualifier** that is used to select an object (or objects) from a larger set of related objects, based upon the qualifier key.
- A *qualifier* is a link value that is unique within the set of links associated with an object in the association.
- In other words, an object and a qualifier value identify a unique object across the associations; they form a composite key.

Overview

More examples:



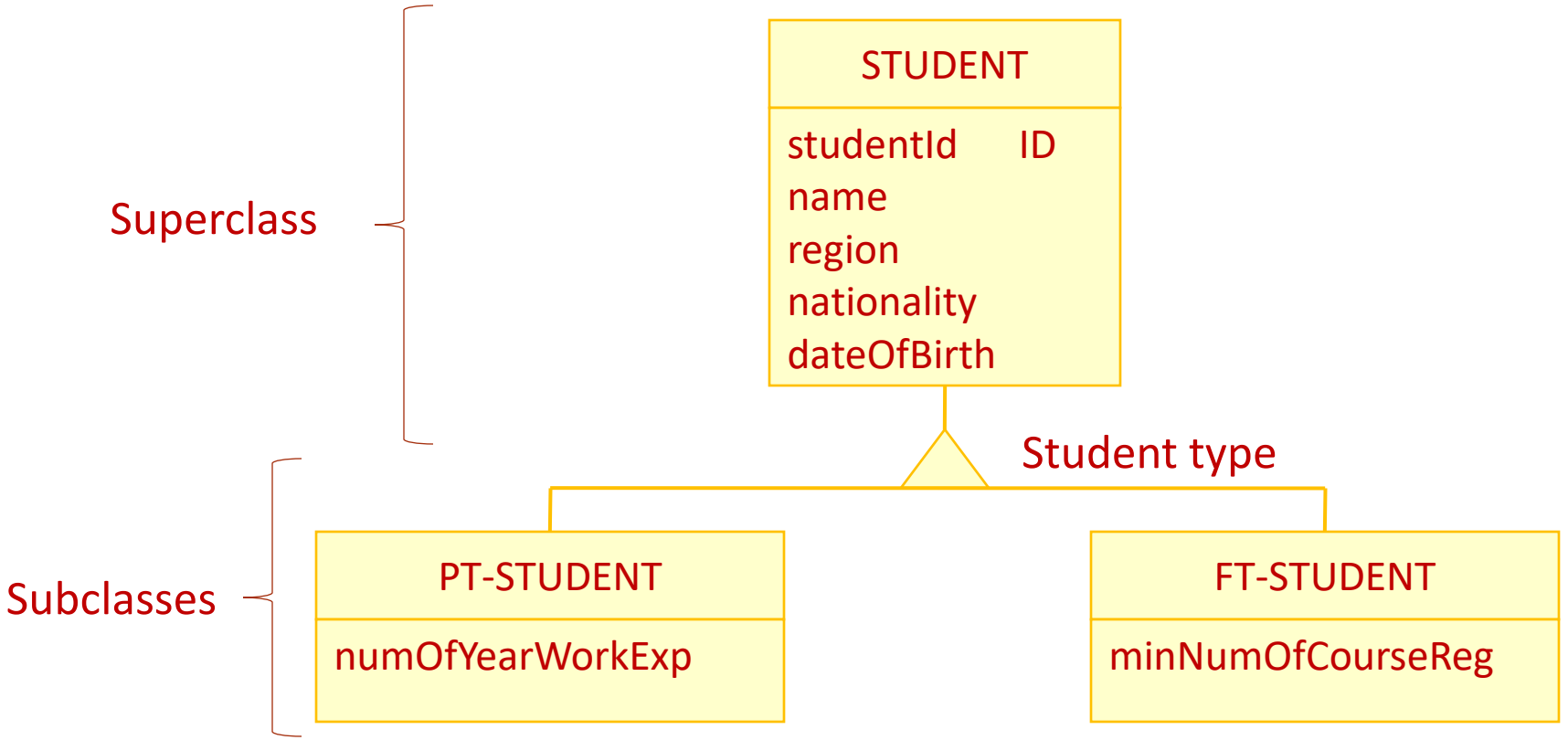
Overview

Generalization/Specialization

- Classes of objects are related in a hierarchy. They can inherit general features from classes that are above them in the hierarchy, known as “superclasses”.
- This concept is also known as generalization.
- Generalization is the taxonomic relationship between a superclass and its subclasses.
- It organizes classes by their similarities and differences, structuring the description of objects.
- All attributes, operations, and associations of a superclass are inherited by all subclasses.

Overview

Diagrammatically, they are depicted as follow:

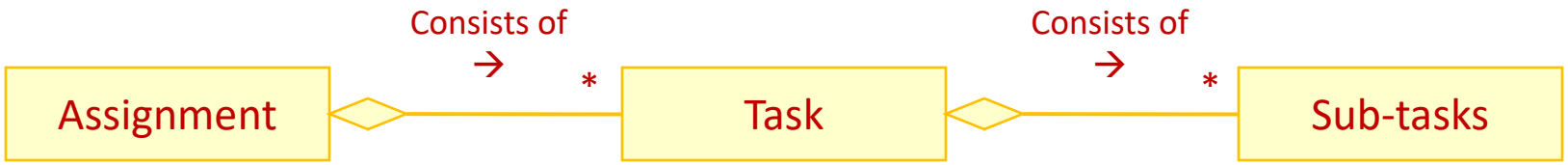


Association – Aggregation/Composition

Composition/Aggregation

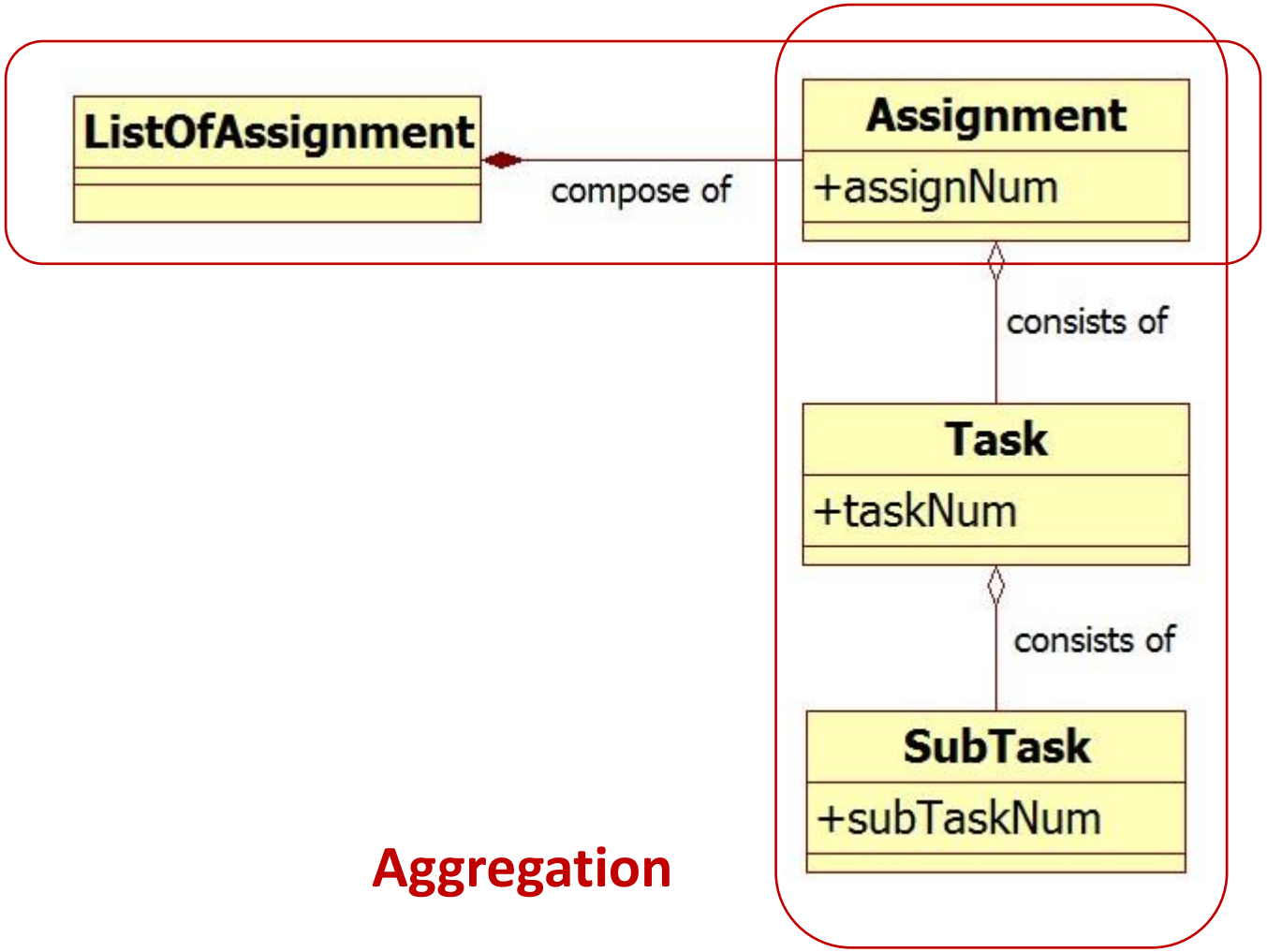
- Composition is the "part-whole" (or compose of) and Aggregation is the "a-part-of" (or consists of) relationship in which objects representing the *components* of something are associated with an object representing the entire *assembly*.

  For example, an assignment consists of many tasks, each of which consists of many sub-tasks.



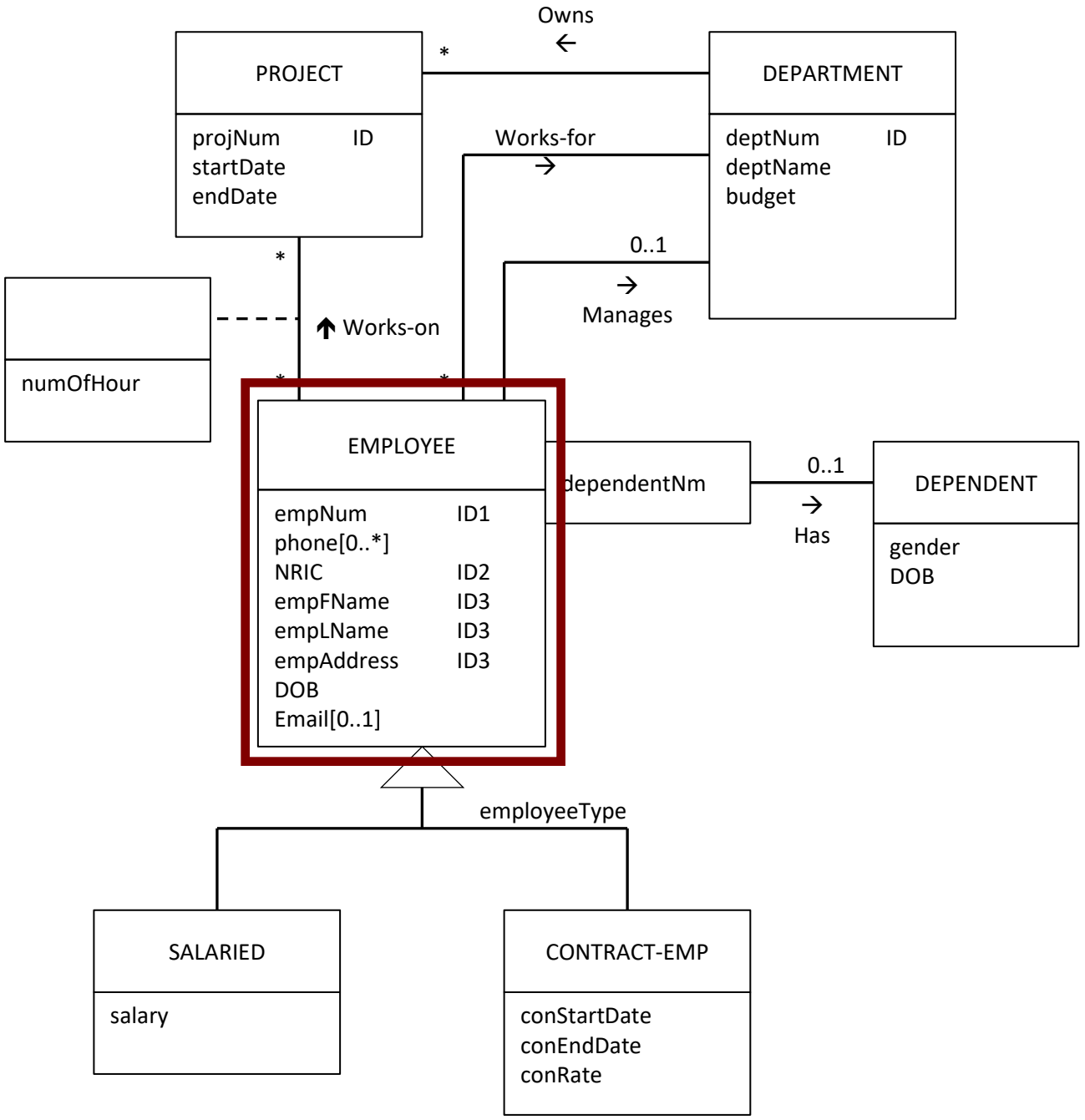
Overview

Composition



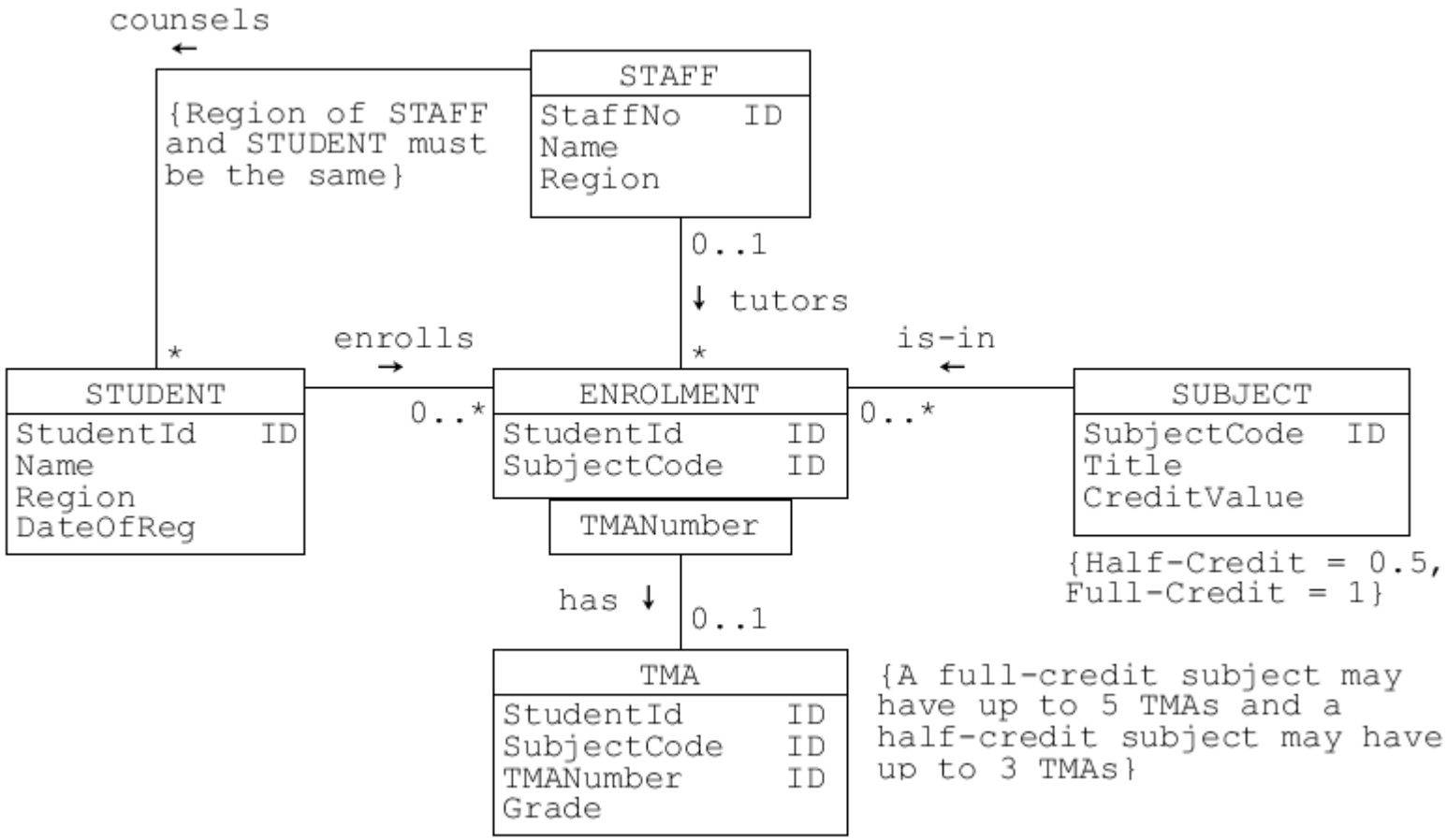
Aggregation

Others


Owns
PROJECT
projNum ID
startDate
endDate
DEPARTMENT
deptNum ID
deptName
budget
Works-for
0..1
Manages
Works-on
numOfHour
EMPLOYEE
empNum ID1
phone[0..*]
NRIC ID2
empFName ID3
empLName ID3
empAddress ID3
DOB
Email[0..1]
dependentNm
0..1
Has
DEPENDENT
gender
DOB
employeeType
SALARIED
salary
CONTRACT-EMP
conStartDate
conEndDate
conRate

Mapping of conceptual model to relational model

How to transform from this (conceptual model) to…



…this Relational Model?

STAFF (StaffNo, Name, Region)
Primary key: StaffNo

STUDENT (StudentId, Name, Region, Counselor)
Primary key: StudentId
Foreign key: Counselor references STAFF (StaffNo)

SUBJECT (SubjectCode, Title, CreditValue)
Primary key: SubjectCode

ENROLMENT (StudentId, SubjectCode, Tutor)
Primary key: (StudentId, SubjectCode)
Foreign key1: StudentId references STUDENT (StudentId)
Foreign key2: SubjectCode references SUBJECT (SubjectCode)
Foreign key3: Tutor references STAFF (StaffNo)

TMA (StudentId, CourseCode, TMANumber, Grade)
Primary key: (StudentId, SubjectCode, TMANumber)
Foreign key: (StudentId, SubjectCode) references ENROLMENT (StudentId, SubjectCode)

Simplification of Conceptual Model

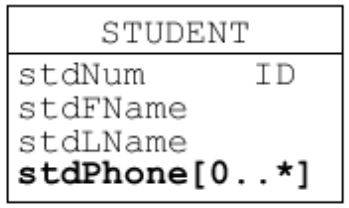
- A simplified conceptual schema is a schema that consists of only:
  - one-to-one or one-to-many associations with classes,
  - **No** multi-value attributes,
  - **No** link-attributes,
  - **No** qualified association and
  - **No** generalization/specialization representation present in the conceptual schema.

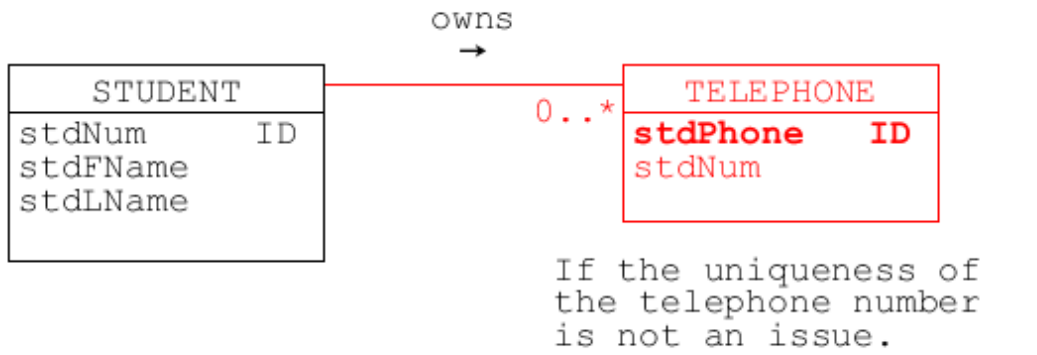
Simplification of Conceptual Model

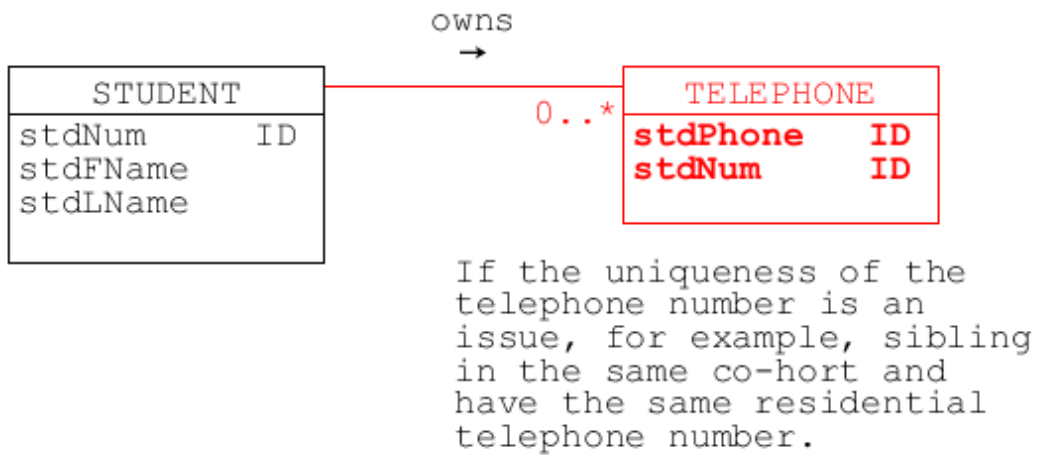
Simplification of multi-value attribute

- For each of multi-value attributes, create a separate relational table,
- Associate the new relational table to the existing relational table using a one-to-many association.
- If the uniqueness of the instances in the new relational table cannot be established, create a composite key by compositing the primary key of the new table to the primary key of the existing table.

Simplification of Conceptual Model





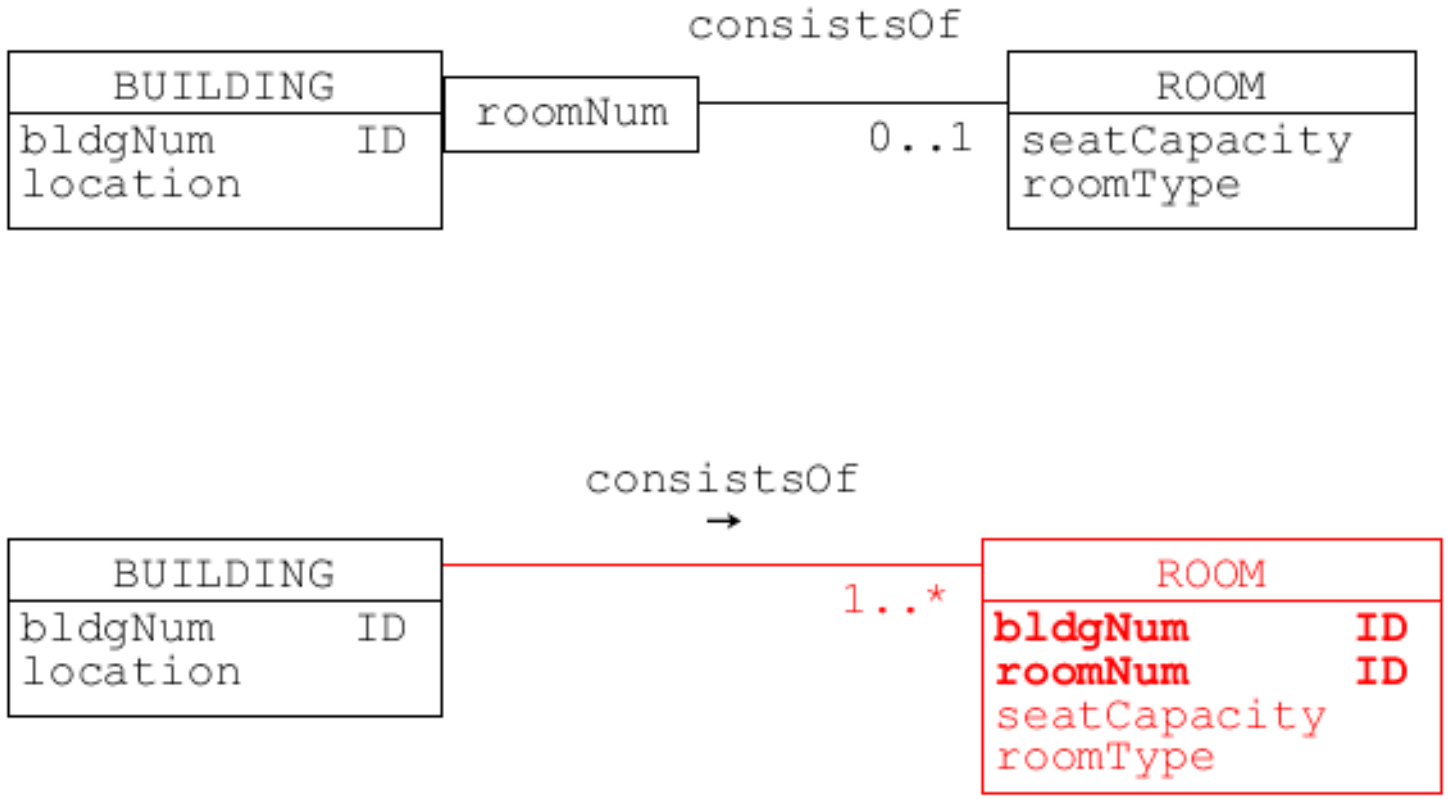


Simplification of Conceptual Model

Simplification of qualified association

- To simplify a qualified association, we simply composite the qualifier to the primary key of the master class it is qualified to uniquely identify the instances in the class.

Simplification of Conceptual Model

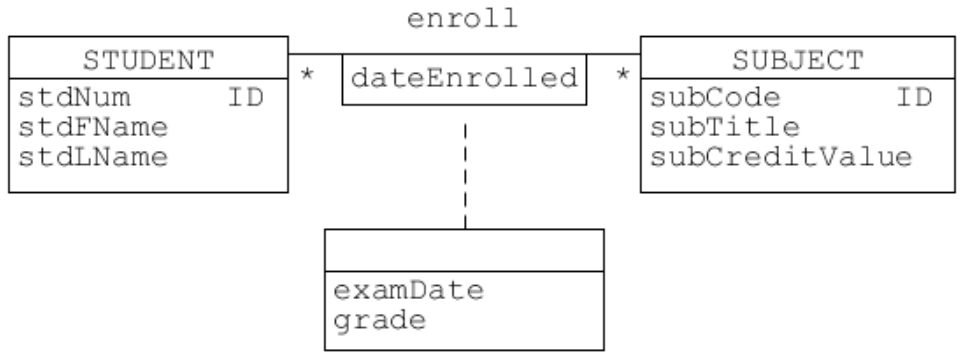


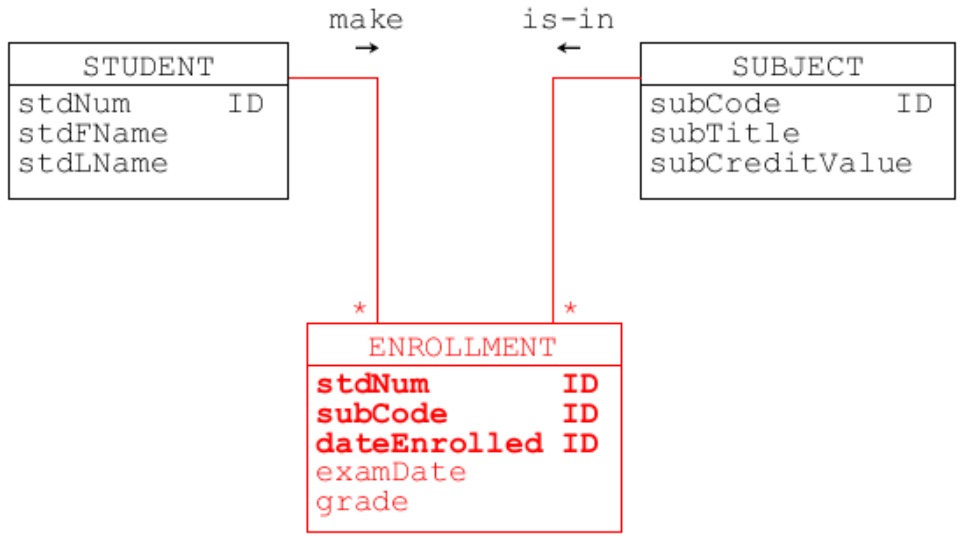
Simplification of Conceptual Model

Simplification of many-to-many association with or without link-attributes

- For each many-to-many association, replace the many-to-many association with an **association class** and **two one-to-many** associations associating the two relational tables to the new association class.
- If there exist association attributes for the many-to-many association, include those attributes into the new association class.
- The primary key of the association class is a **composite key** consisting of the primary keys from the classes it associates plus the qualifier if any.

Simplification of Conceptual Model





Simplification of Conceptual Model

Simplification of Specialization or Generalization

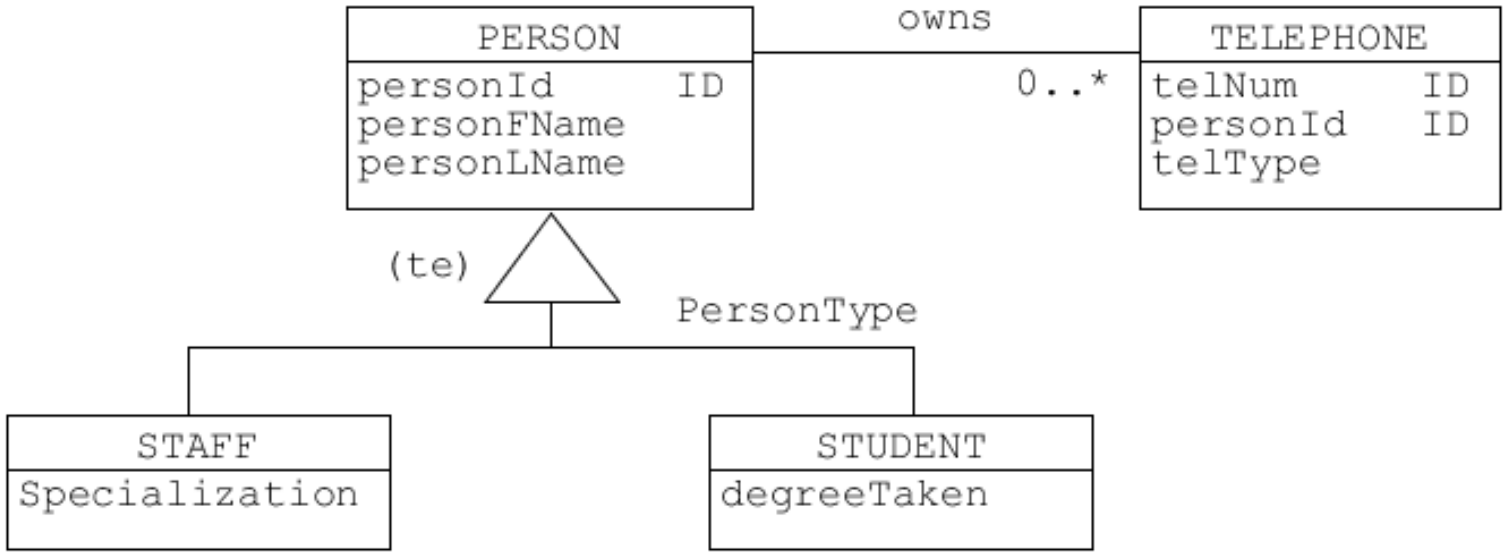
- Specialization or generalization can be simplified by first converting each specialization with m subclasses $\{S_1, S_2, \cdots, S_m\}$ and generalising superclass C, where the attributes of the superclass C are $\{k, a_1, a_2, \cdots, a_n\}$ and k is the primary key, into relational tables using one of the three following options:
  i. Association with multiple relational tables of superclass and subclasses.
  ii. Migrating of the superclass attributes down to subclasses, and hence creating a horizontal segmentation of the data.
  iii. Migrating of the subclasses attributes to the superclass and include the discriminator.

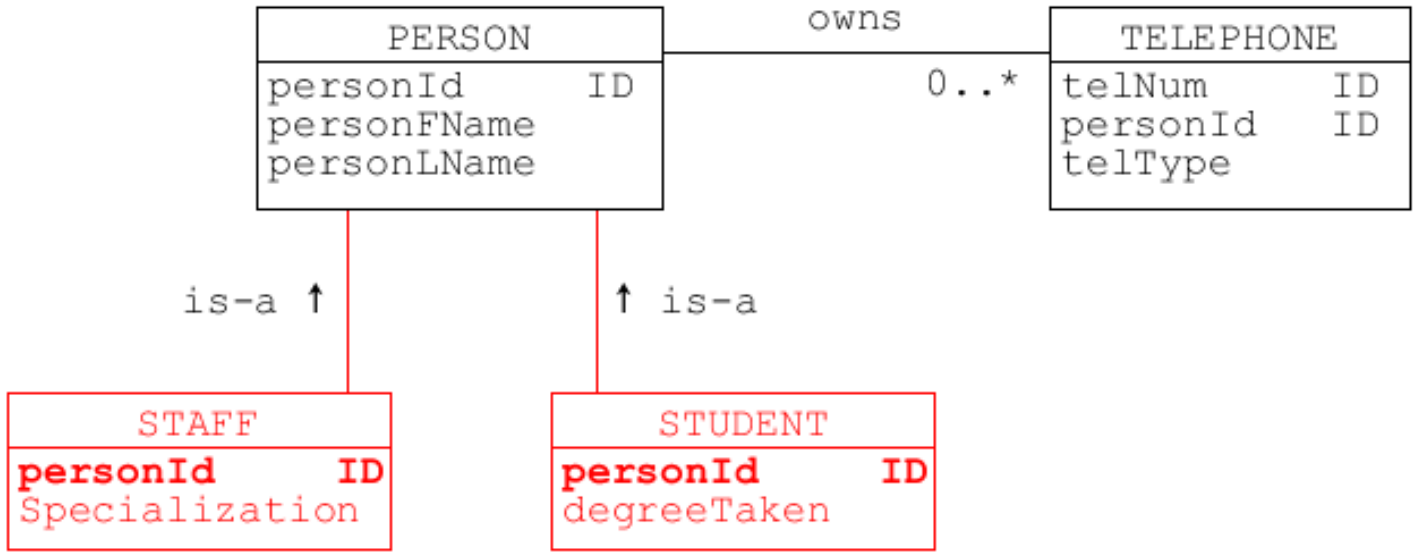
Simplification of Conceptual Model

- **Resolving generalization using association –** transform the generalization into one-to-one association. For each of the subclasses, associate it to the superclass.

Simplification of Conceptual Model

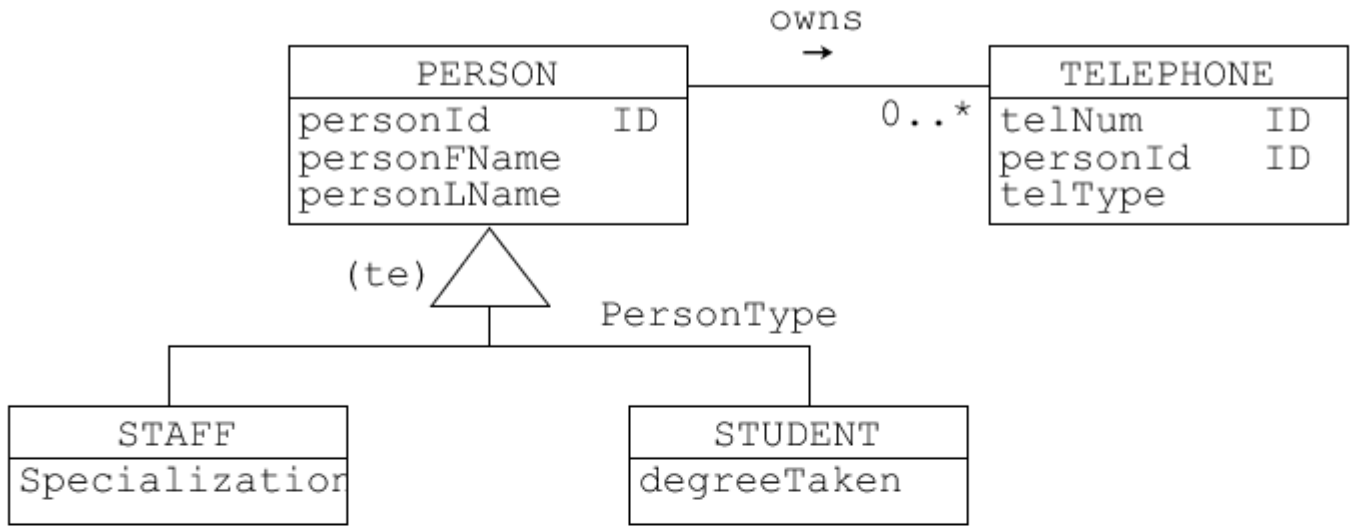
- **Resolving generalization using association –** transform the generalization into one-to-one association. For each of the subclasses, associate it to the superclass.

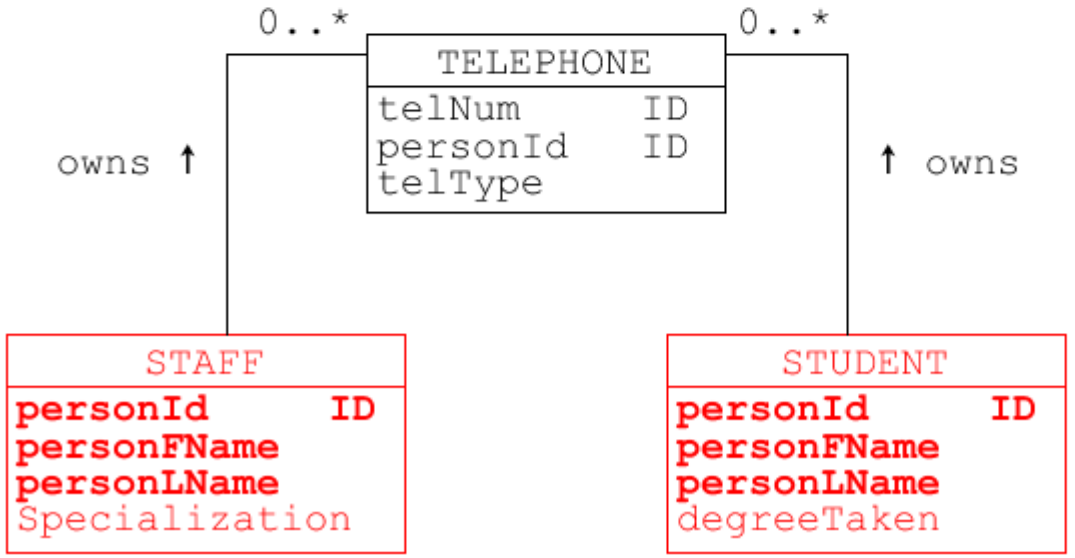




Simplification of Conceptual Model

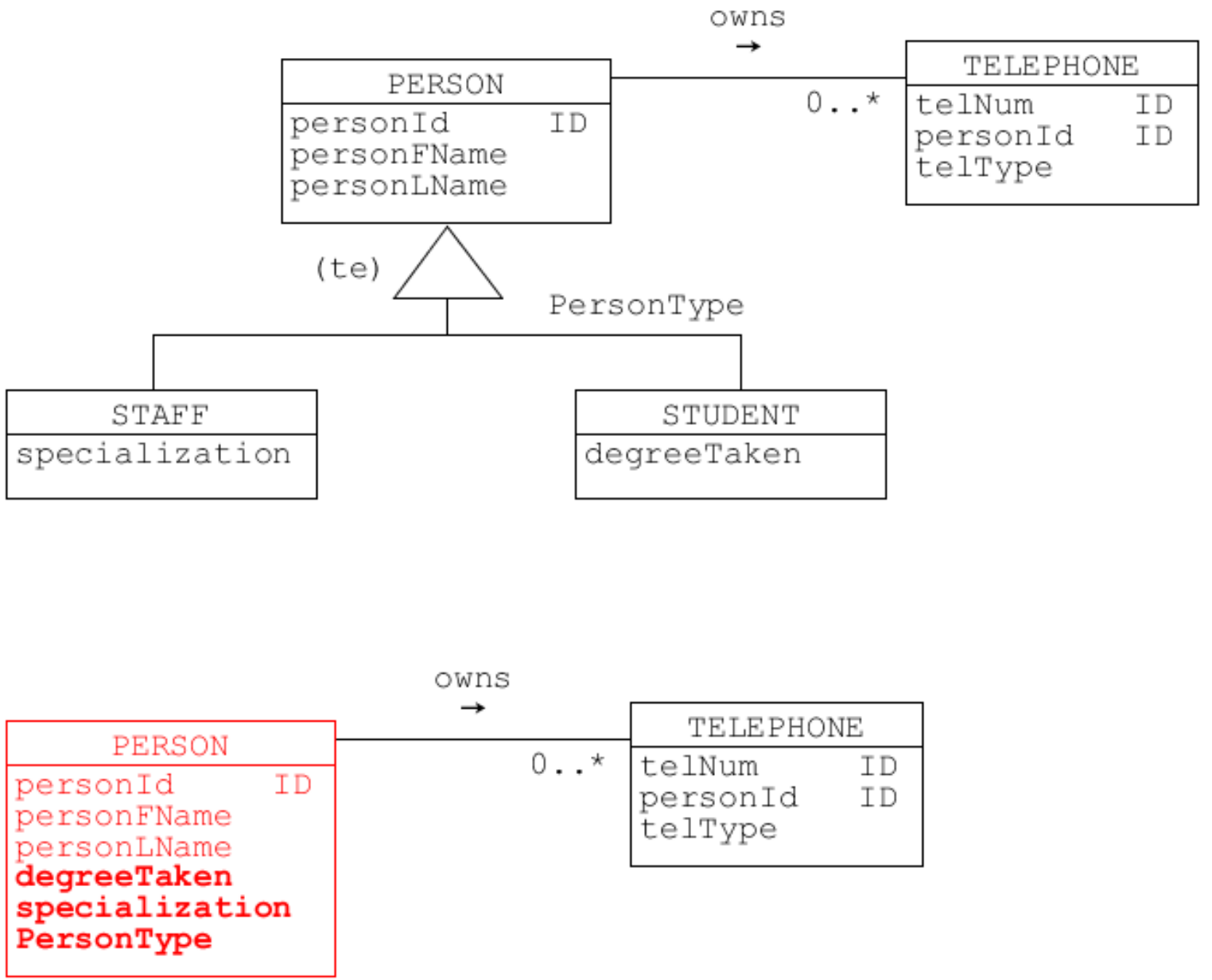
- **Resolving generalization using subclasses -** Migrating of the superclass attributes down to subclasses, and hence creating a horizontal segmentation of the data.





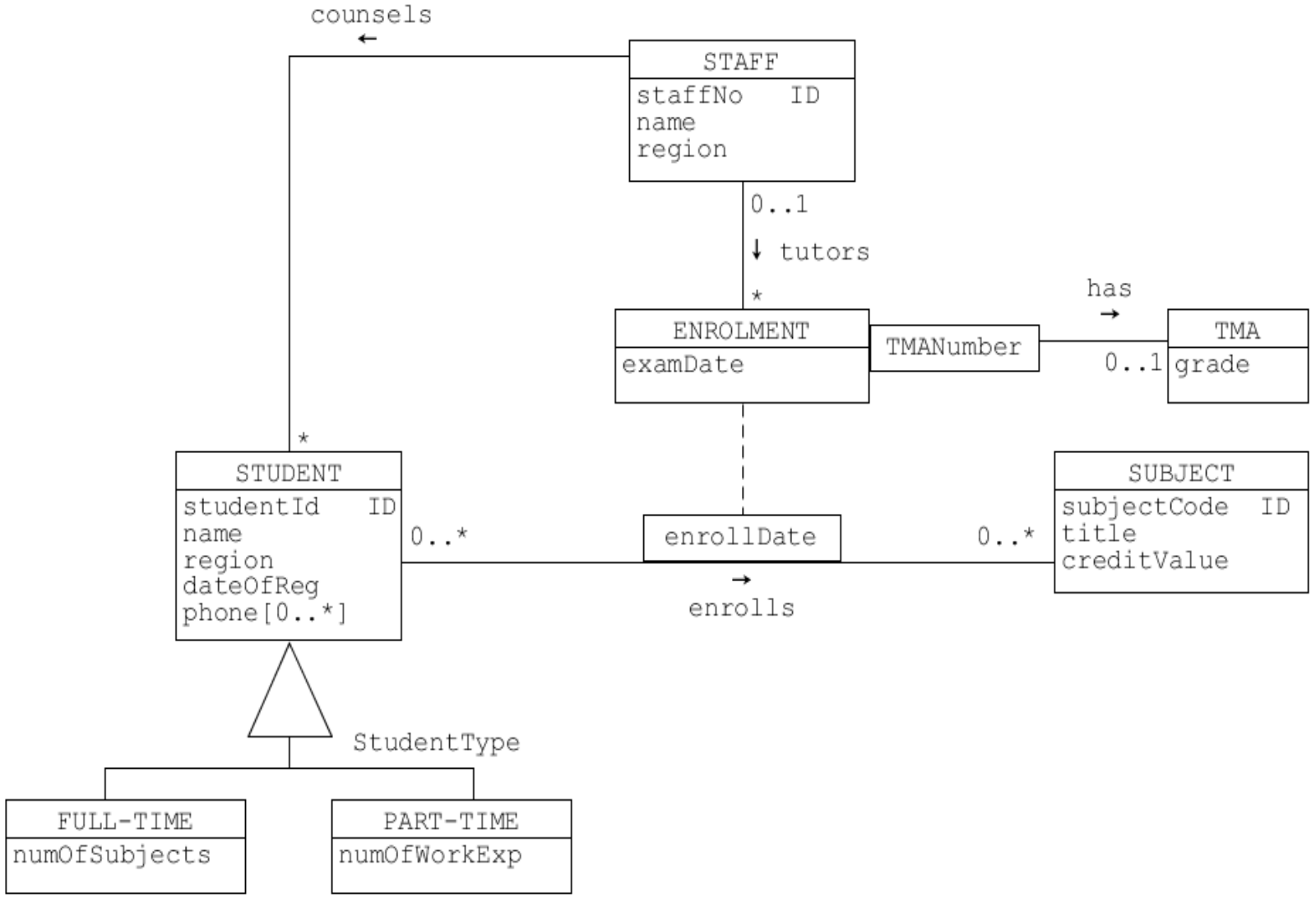
Simplification of Conceptual Model

- **Resolving generalization using superclass –** in this method, we migrate all the subclasses attributes to the superclass, including the discriminator.



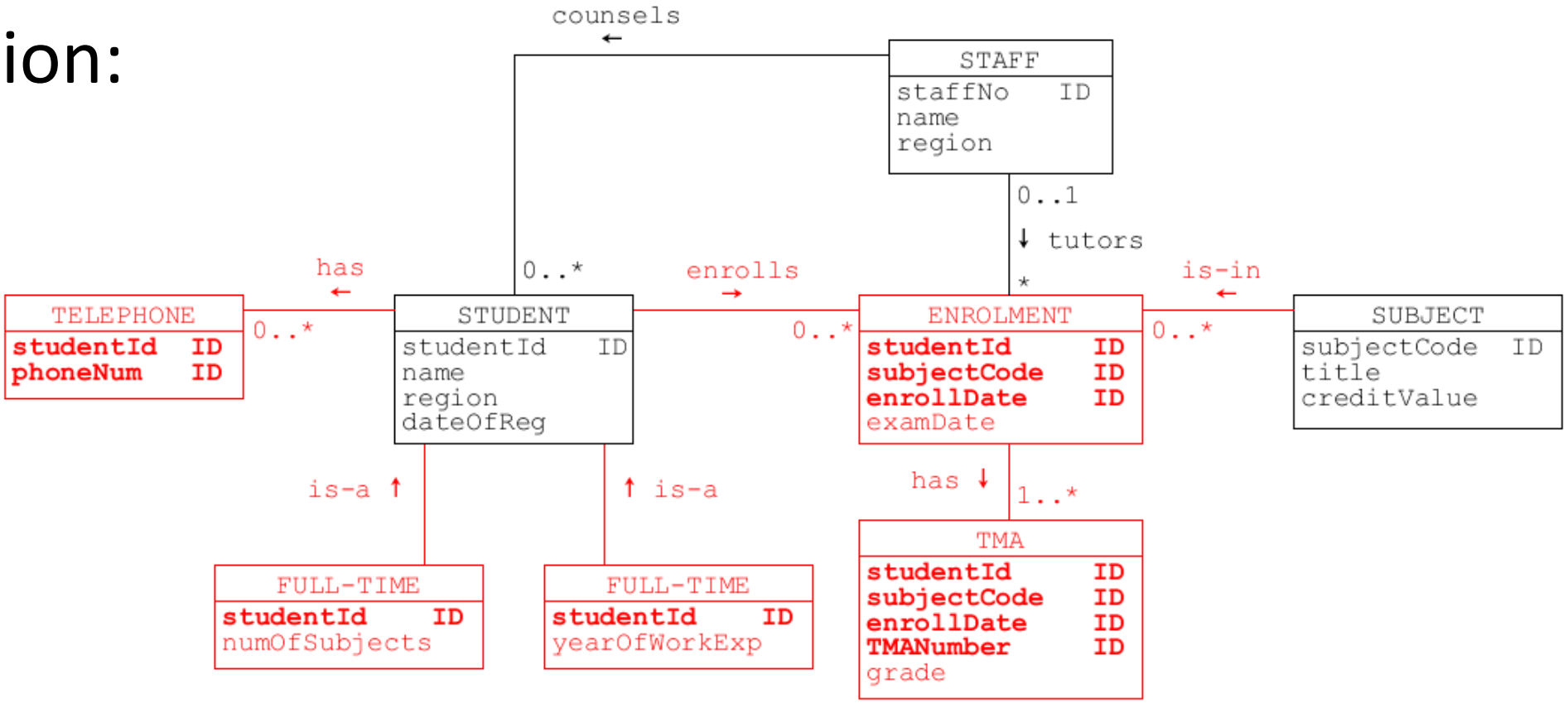
Simplification of Conceptual Model

Exercise: Map the conceptual schema shown here into a relational schema.



Simplification of Conceptual Model

Solution:

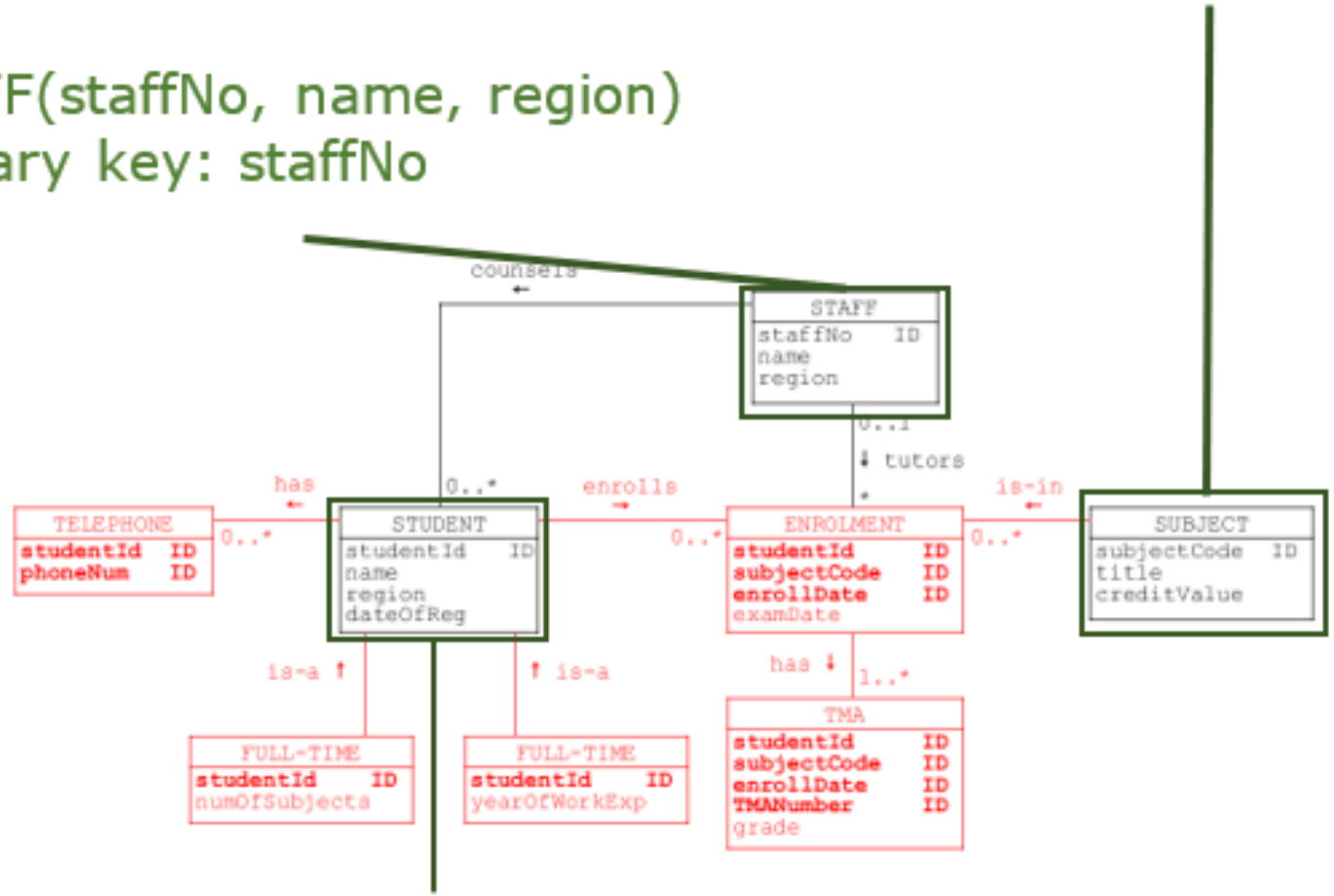


Simplification of Conceptual Model

SUBJECT(subjectCode, title, creditValue)
Primary key: subjectCode

STAFF(staffNo, name, region)
Primary key: staffNo

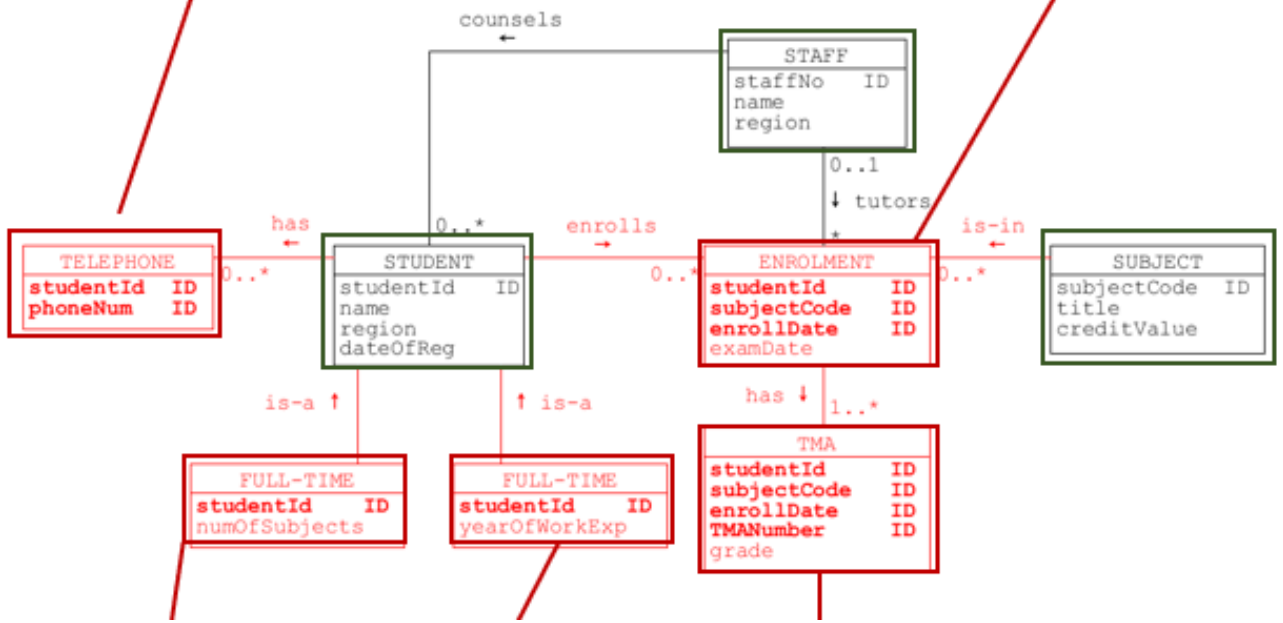
- Next, we map the classes.

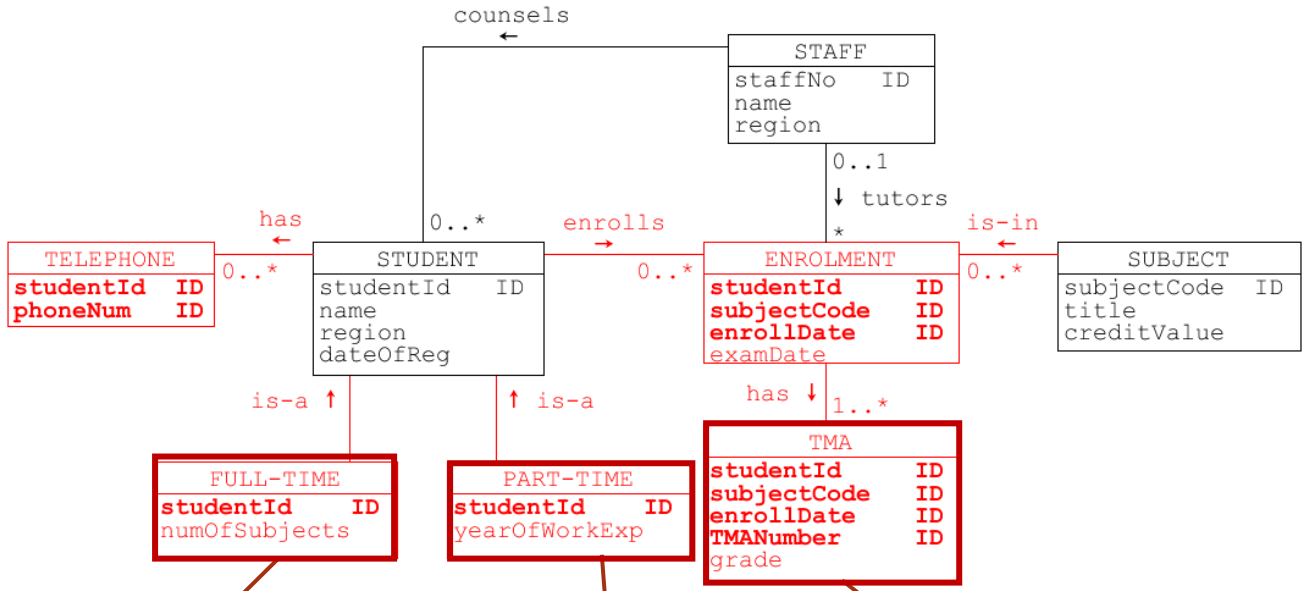


STUDENT(studentId, name, region, dateOfReg)
Primary key: studentId

TELEPHONE(studentId, phoneNum)
Primary key: (studentId, phoneNum)

ENROLMENT(studentId, subjectCode, enrolDate, examDate)
Primary key: (studentId, subjectCode, enroldate)





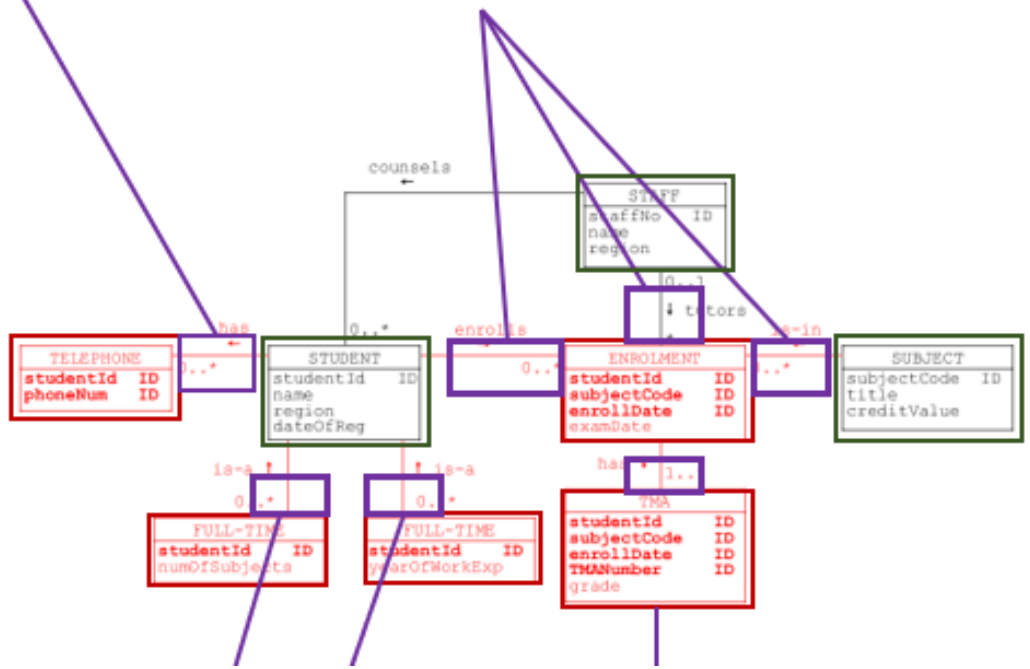
FULL-TIME(studentId, numOfSubjects)
Primary key: (studentId)

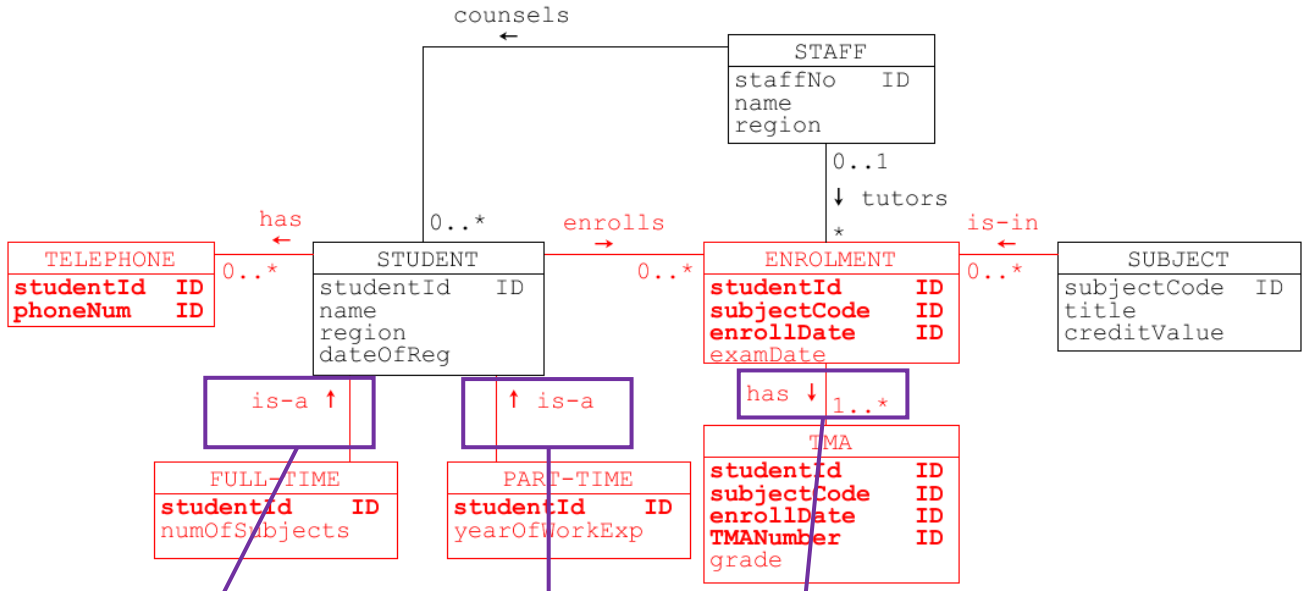
TMA(studentId, subjectCode, enrolDate, TMANumber, grade)
Primary key: (studentId, subjectCode, enrolDate)

PART-TIME(studentId, yearOfWorkExp)
Primary key: (studentId)

TELEPHONE(studentId, phoneNum)
Primary key: (studentId, phoneNum)
Foreign key: studentId references STUDENT(studentId)

ENROLMENT(studentId, subjectCode, enrolDate, examDate, tutor)
Primary key: (studentId, subjectCode, enroldate)
Foreign key1: studentId references STUDENT(studentId)
Foreign key2: subjectCode references SUBJECT(subjectCode)
Foreign key3: tutor references STAFF(stafNo)

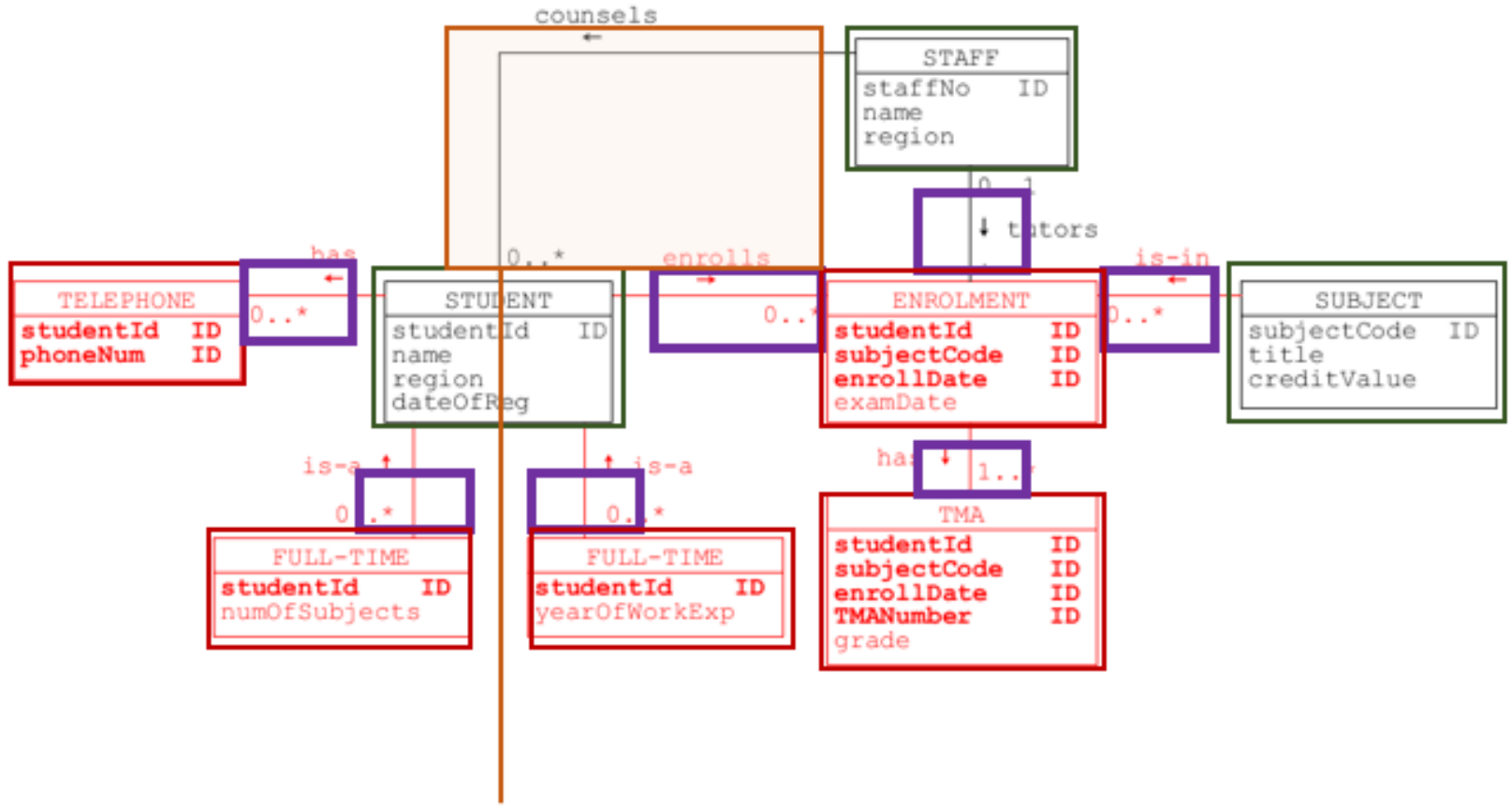




FULL-TIME(studentId, numOfSubjects)
Primary key: (studentId)
Foreign key: (studentId) references STUDENT(studentId)

TMA(studentId, subjectCode, enrolDate, TMANumber, grade)
Primary key: (studentId, subjectCode, enrolDate)
Foreign key: (studentId, subjectCode, enrollDate) references
ENROLMENT(studentId, subjectCode, enrollDate)

PART-TIME(studentId, yearOfWorkExp)
Primary key: (studentId)
Foreign key: (studentId) references STUDENT(studentId)



STUDENT(studentId, name, region, dateOfReg, counsellor)

Primary key: studentId

Foreign key: counsellor references STAFF(staffNo)

Putting together

STAFF(staffNo, name, region)

Primary key: staffNo

SUBJECT(subjectCode, title, creditValue)

Primary key: subjectCode

STUDENT(studentId, name, region, dateOfReg, counsellor)

Primary key: studentId

Foreign key: counsellor references STAFF(staffNo)

Putting together

TELEPHONE(studentId, phoneNum)

Primary key: (studentId, phoneNum)

Foreign key: studentId references STUDENT(studentId)

ENROLMENT(studentId, subjectCode, enrolDate, examDate, tutor)

Primary key: (studentId, subjectCode, enrolDate)

Foreign key1: studentId references STUDENT(studentId)

Foreign key2: subjectCode references SUBJECT(subjectCode)

Foreign key3: tutor references STAFF(staffNo)

Putting together

TMA(studentId, subjectCode, enrolDate, TMANumber, examDate, grade)

Primary key: (studentId, subjectCode, enroldate)

Foreign key: (studentId, subjectCode, enrolDate) references ENROLMENT (studentId, subjectCode, enrolDate)

Putting together

FULLTIME(studentId, numOfSubjects)

Primary key: (studentId)

Foreign key: studentId references STUDENT(studentId)

PARTTIME(studentId, yearOfWorkExp)

Primary key: (studentId)

Foreign key: studentId reference STUDENT(studentId)