Lecture 2 Entity-Relationship Model Modeling

Outline

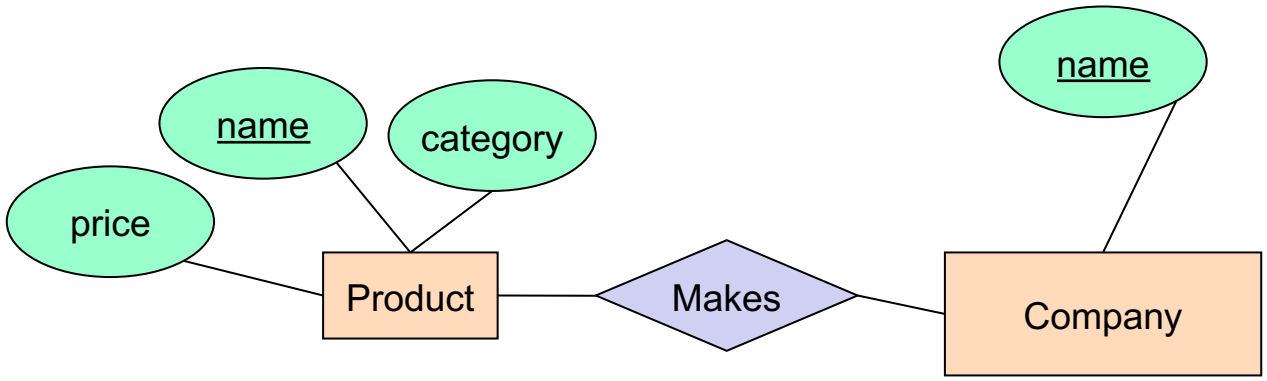
- Design Process
- Entity Sets
- Attributes
- Relationship Sets

Design Process

- In general, the goal of the relational database design is to create a database in a specific database management system that allows us to
  - store all information that we want,
  - minimize unnecessary redundancies, and
  - let users search information easily.
- However, database design for an application is a complex task.
- It is split into multiple phases.

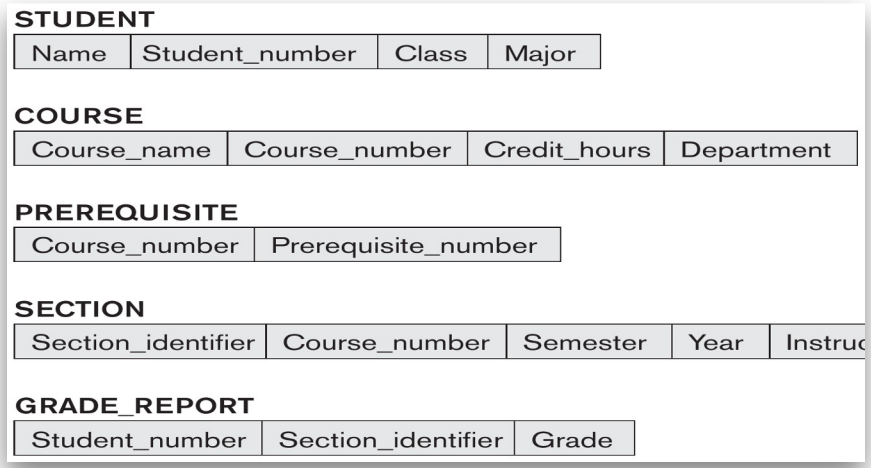
Design Process

- The first phase of the design process is ***conceptual-design***.
- At this stage, designers focus on describing data and their relationships.
- The ***entity-relationship model*** (ER model) is introduced in this course.
- The outcome of this phase is an ***ER diagram***.
- It provides a graphic representation of the database design.



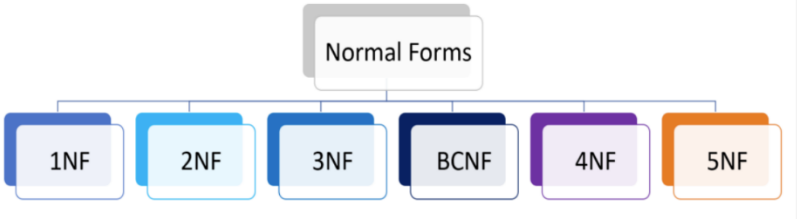
Design Process

- The next phase is to convert an ER diagram to the implementation data model of the database system that will be used.
- In this course, we will introduce the relational data model.
- Thus, ***relational schemas*** will be produced after this phase.
- In general, relational schemas are some descriptions for the data which is modeled in an application.
- The relational schemas of a database are called logical view.
- So, this phase is the logical-design phase.



Design Process

- After the second phase, the designer has several relational schemas.
- However, the relational schemas may not be perfect and contain redundancies, because the ER diagram is very subjective.
- Then, the next phase is optimizing the logical design using ***functional dependencies*** and ***normal forms***.
- The optimization will be introduced in the chapter Database Design.



Design Process

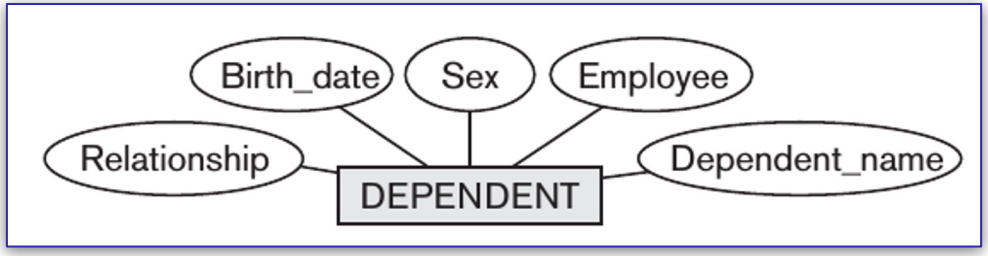
- The final phase will be the ***physical-design phase***.
- Designers will implement the optimized relational schemas in the database management system using a particular ***data definition language***.
- **SQL** will be introduced in this course.
- After the implementation, the design of a database is finished.
- Users can load data into the database.
- Then, users can manipulate data and develop database applications.

Entity Set

- The entity-relationship model contains three basic concepts:
  - entity sets,
  - relationship sets, and
  - attributes.
- An ***entity*** is a “thing” or “object” in the real world, which is distinguishable from all other objects.
  - E.g.: the STUDENT John Smith; the CST DEPARTMENT
- An ***entity set*** is the class or type of objects in our model.
- For example,
  - The student David is an entity.
  - The student with ID is 2079 is another entity.
  - The entity set “student” contains all students.

Attribute

- An entity can be described by a set of properties.
- Each property is an ***attribute*** of the entity.
- A set of attributes describes and distinguishes the entities in the same entity set.



- One entity can have different attributes in different models for different applications.
- E.g. in SAO's database, room number is an attribute for student, while AR database is more interested in his GPA instead.

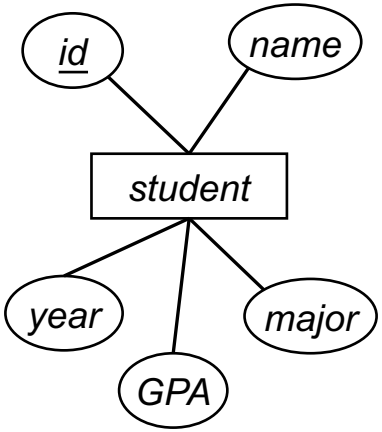
Attribute

- Each entity has a ***value*** for each of its attributes.
- A particular student entity can have values 2079 for ID, $David$ for name, 2 for year, and CST for major.

- The ***Domain*** of an attribute is the set of all possible values of the attribute.
- The domain of the attribute major consists of all majors in UIC (CST, FST, ACCT, etc.)

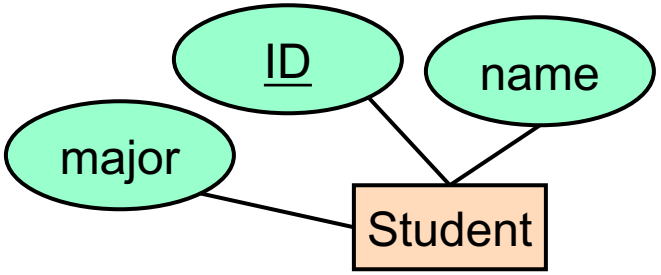
Basic ER Features

- In ER diagrams,
  - ***rectangles*** represent entity sets;
  - ***ellipses*** represent attributes;
  - ***keys*** are underlined; and
  - ***lines*** link attributes to entity sets.
- For example, the student entity set is modeled as follows.



Key

- To distinguish two entities, we compare the values.
- If two entities have different values for a same attribute, then they are different entities.
- Equivalently, if two entities are identical, then they agree on the values for all attributes.
- E.g. David and Goliath are both CST students.
  - The same value for major attribute
  - Different values for name attribute
  - They are different students.

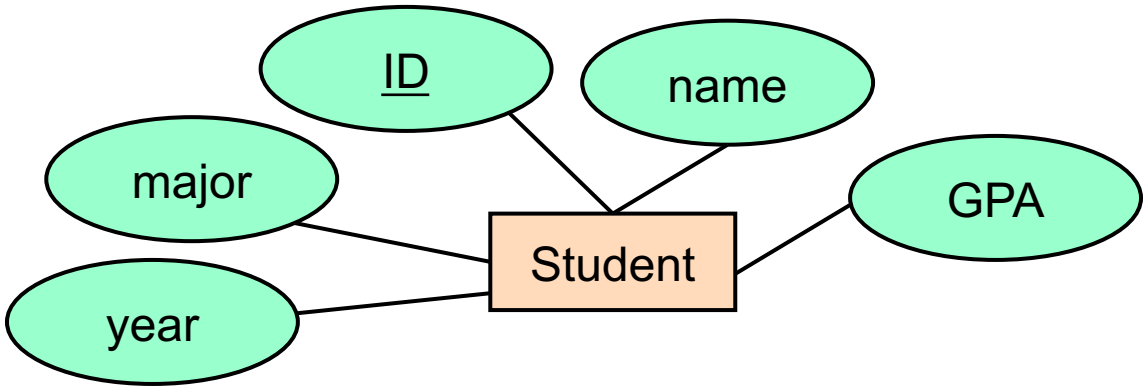


Key

- To test if two entities are identical, we can check all attributes.
- But this is not efficient.
- In most cases, if two entities have the same value for one or some special attributes, then we can sufficiently claim that the two entities are the same.
- The set of special attribute(s) is called ***key***.
- Formally, a key of an entity set is a set of attributes that can ***uniquely identify*** the entities.
- Two entities are identical **if and only if** they have the same value for the key.

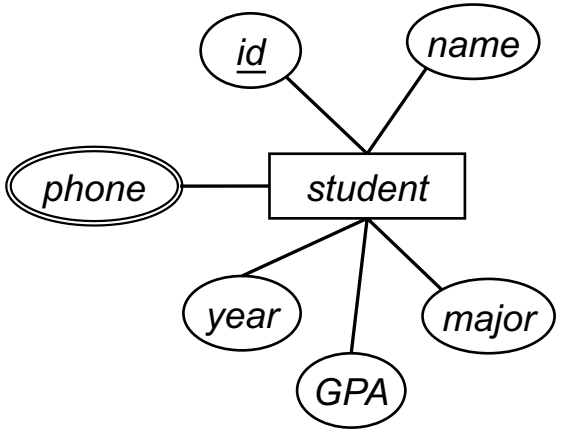
Key

- For example, each student is described by ID, name, year, major, and GPA.
- In the real-world, two students may have the same name, year, major, and even GPA. But their student IDs have to be different.
- The student ID can distinguish different students.



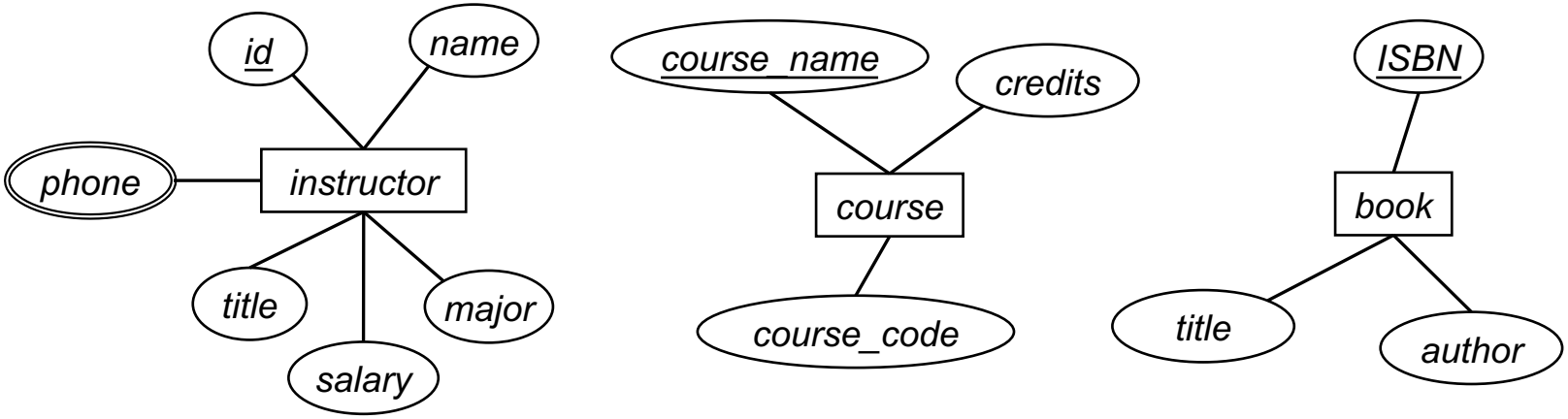
Multi-valued Attribute

- Suppose that we also want to model students' phone number.
- It is possible that one student may have multiple phone numbers.
- Thus, the phone number of a student is a ***multi-valued*** attribute, denoted by ***double ellipses***.



Example

- For more examples, instructor, course, and book are modeled in the same way.

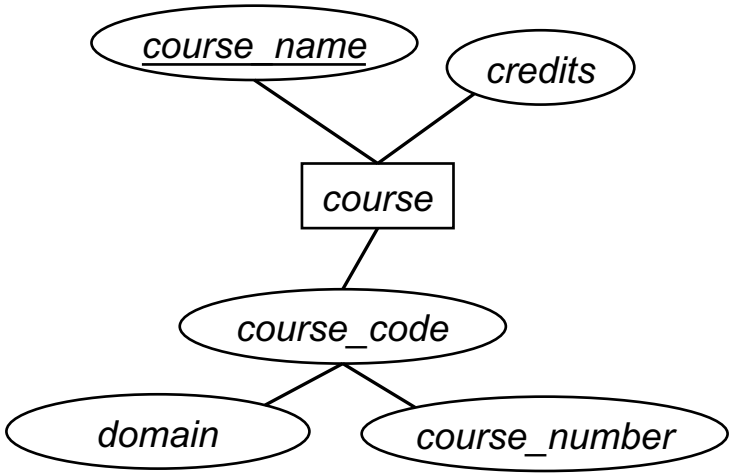


Composite Attribute

- One may ask “why the course code is not selected as a key?”
- The reason is that a course code is not ***atomic***.
- An attribute is atomic if each value of the attribute has only one unit of information.
- If an attribute is not atomic, it is a ***composite attribute***.
- For example, the course code for this database course is “$COMP3013$”.
  - “$COMP$”: the course is in the domain computer science.
  - “3013”: the course number.

Composite Attribute

- Thus, the attribute course code is decomposed into domain (offering unit) and course number.

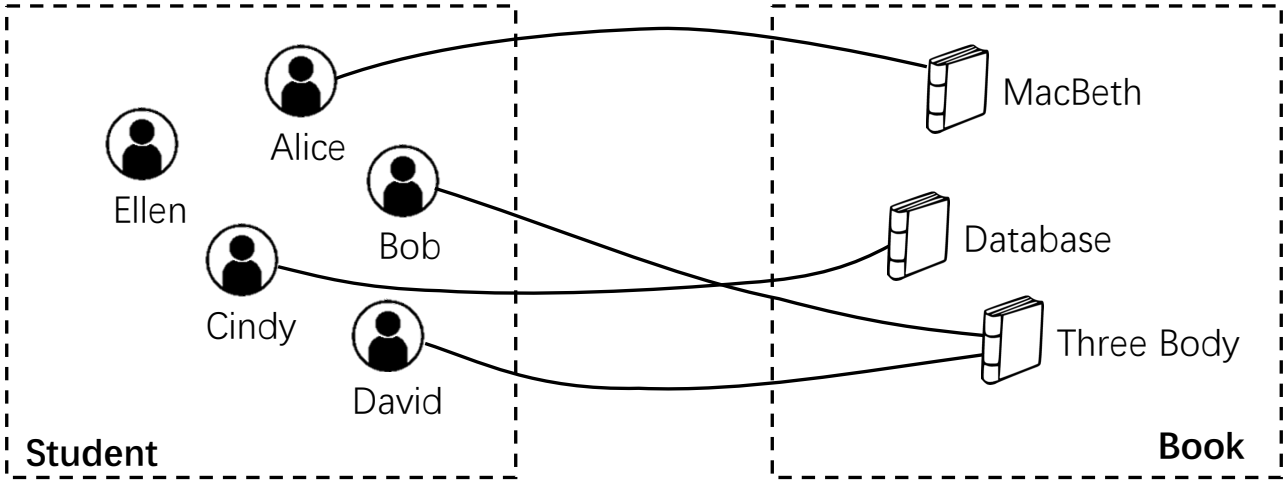


Relationship

- The users of a database also want to model the associations among multiple entities.
- For example, the student *"David"* borrows the book *"Three Body"*.
- The student and the book are two entities. And "borrow" is an association between them.
- The association is called ***relationship***.
- To present a relationship, we can put the entities which participate the relationship as a tuple.
- The above example can be ("*David*", "*Three Body*").
- Note that "*David*" and "*Three Body*" are not values of two attributes. They are two entities.

Relationship Set

- A ***relationship set*** is a set of relationships of the same type.
- For example, the relationship set “borrow” describes which student borrows which book.



- The set of all links is the relationship set.

Relationship Set

- The above relationship set can also be presented as a set in mathematics.

$$borrow = \{(Alice, MacBeth), (Bob, Three\ Body)$$
$$(Cindy, Database), (David, Three\ Body)\}$$

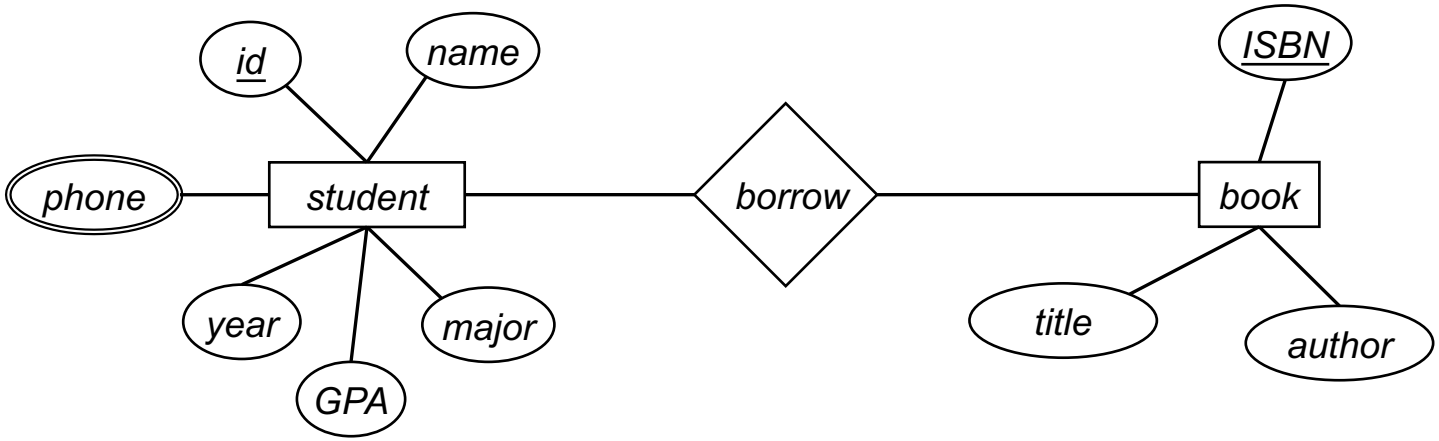
- Note that in mathematics, sets do not allow duplications. So, the “borrow” relationship set is same as

$$borrow = \{(Alice, MacBeth), (Bob, Three\ Body)$$
$$(Cindy, Database), (David, Three\ Body)$$
$$(Cindy, Database), (David, Three\ Body)\}$$

- This is also applied to entity sets.

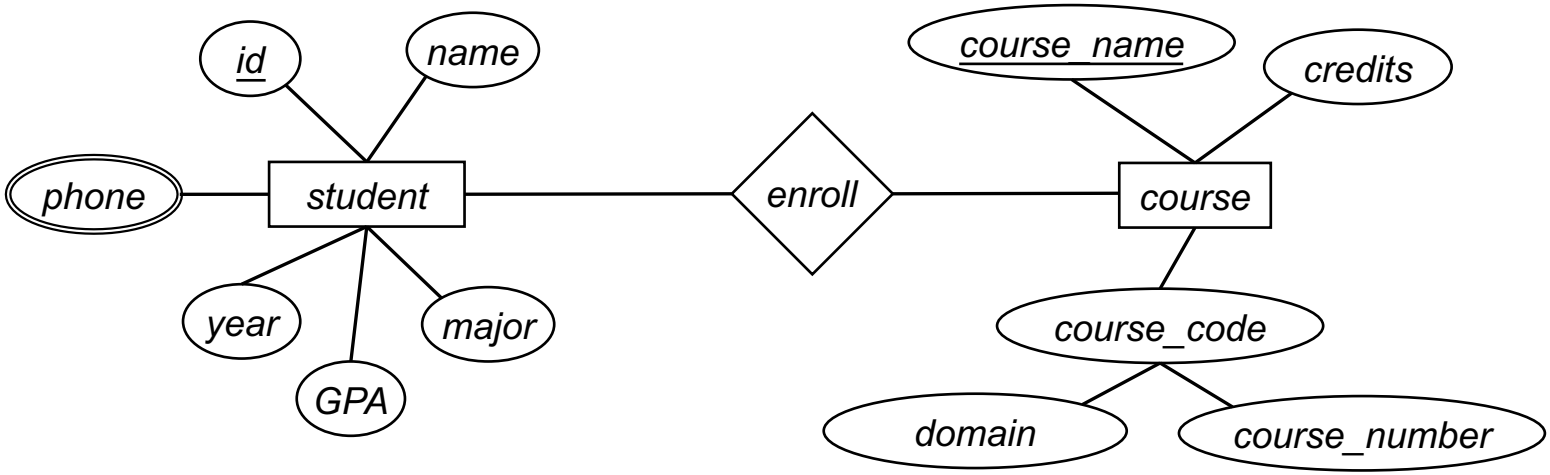
Relationship Set

- In ER diagrams, a relationship set is denoted by a ***diamond***.
- The previous example “students borrow books” can be modeled as



Relationship Set

- Here is another example. Suppose we want to model “Some students are enrolled in some courses.”
- “Enroll” is the relationship associating the two entity sets.

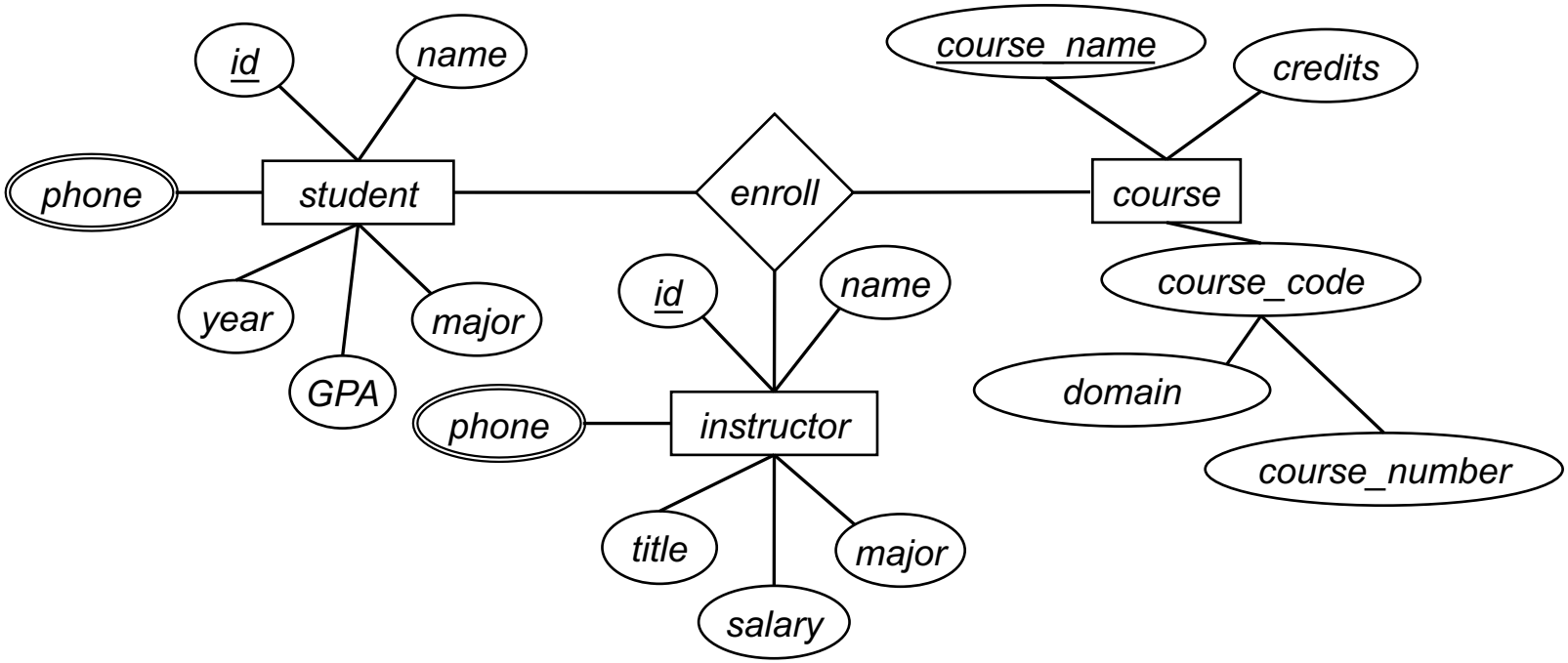


Multi-ary Relationship

- If a relationship associates n entities, this relationship is ***n*-ary**.
- n is the ***degree*** of the relationship
- If $n = 2$, the relationship is ***binary***.
- If $n = 3$, the relationship is ***ternary***.
- Theoretically, n can be any positive integer. But in this course, n is at most 3.

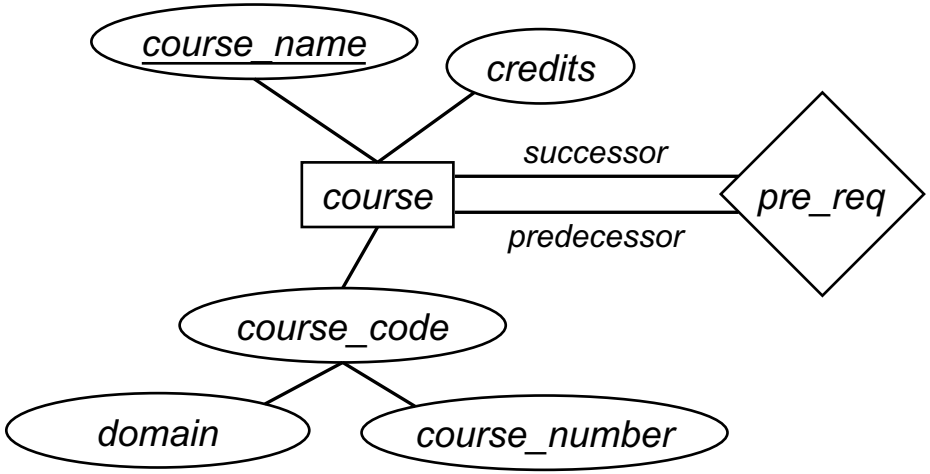
Multi-ary Relationship

- For the following example, a ternary relationship is reasonable.
- This example models that some students are enrolled in some courses which are instructed by some teachers.



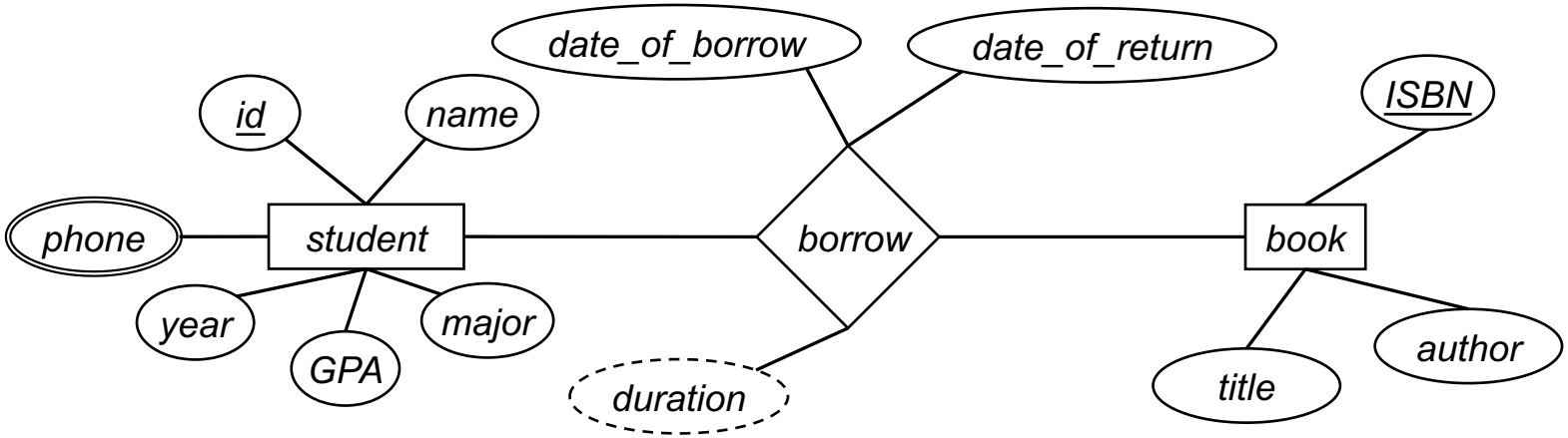
Roles

- Sometimes multiple entities of the same type can participate a same relationship.
- Suppose we want to express “some courses are the prerequisite of some other courses”.
- For example, before taking the database course, one must pass the C program course.
- To model this example, ER diagram allows an entity set to link with a relationship set multiple times.
- Roles are written in text, to express how the entities are participating the relationship.



Attributes for Relationship Sets

- Sometimes people are also interested in some information about relationships.
- In the “students borrow books” example, we also want to know when the book is borrowed, when the book is returned, and how long the book is kept by the student.
- This information does not belong to students or books. It is about the association.
- Thus, in this case, the relationship set can also have some attributes.

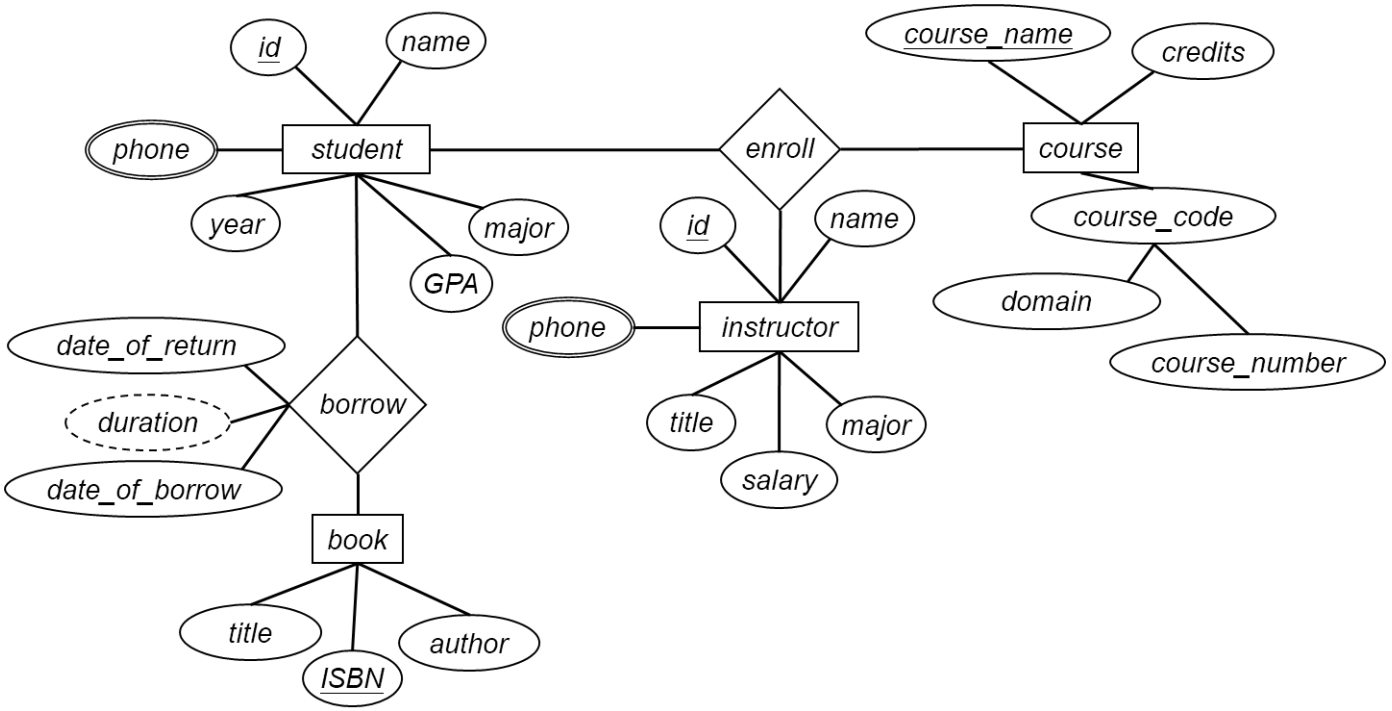


Derived Attributes

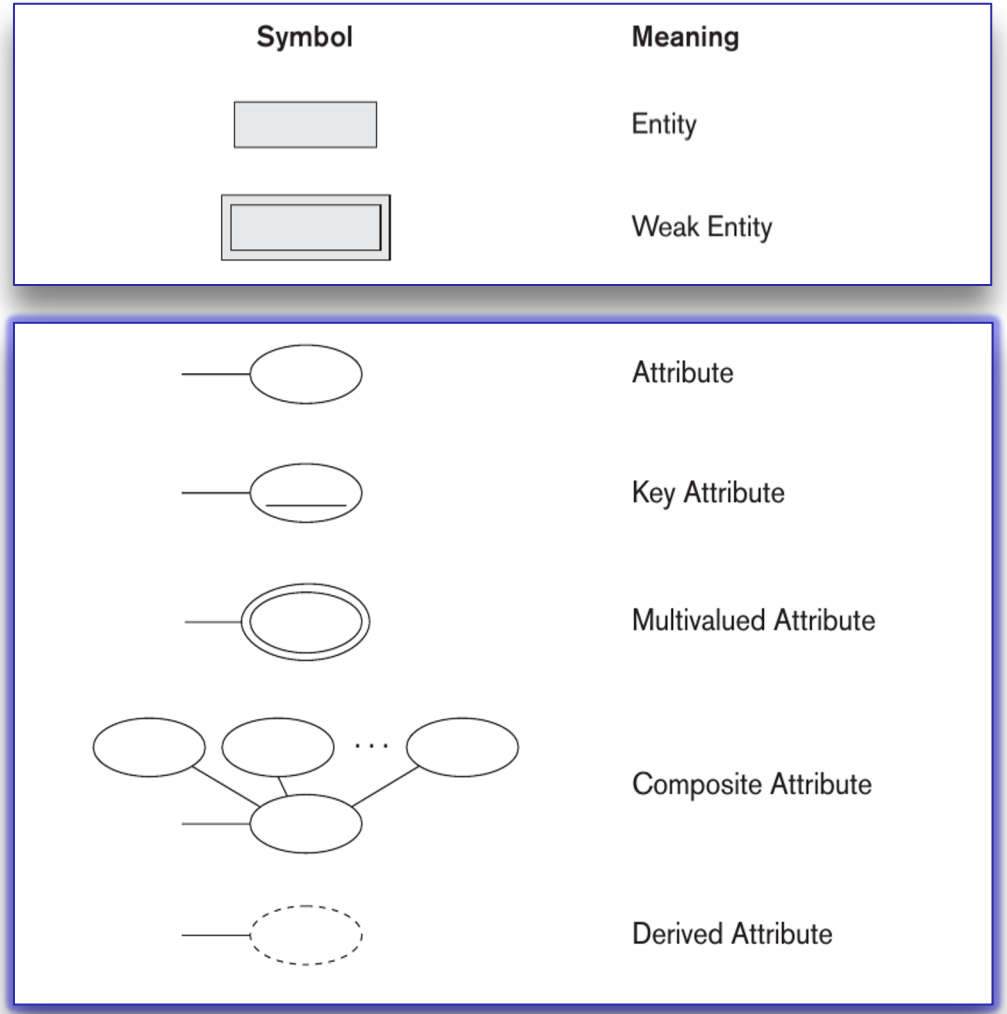
- In the previous example, the attribute duration is in a ***dashed ellipse*** because it is a ***derived attribute***.
- If one knows the date of borrow and the date of return, then the duration can be calculated from the two values.

Example

- Summarizing the examples.
- Sometimes designers have to reallocate the positions of some components of the ER diagram to make it beautiful.



Summary


Symbol
Meaning
Entity
Weak Entity
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute

Exercises

- Based on the ER diagram on page 29, model the following features.
  - Programs, which have program codes, program names, and the division that each program belongs to.
  - Students have majors.
  - Instructors work for some programs.
  - Every program has a program director, who is also an instructor.
  - Courses are offered by programs.

End of Lecture 2