DSAA 5012
Advanced Database Management for Data Science

LECTURE 2
ENTITY-RELATIONSHIP (E-R) MODEL AND DATABASE DESIGN

E-R MODEL & DB DESIGN: OUTLINE

Database Design Process

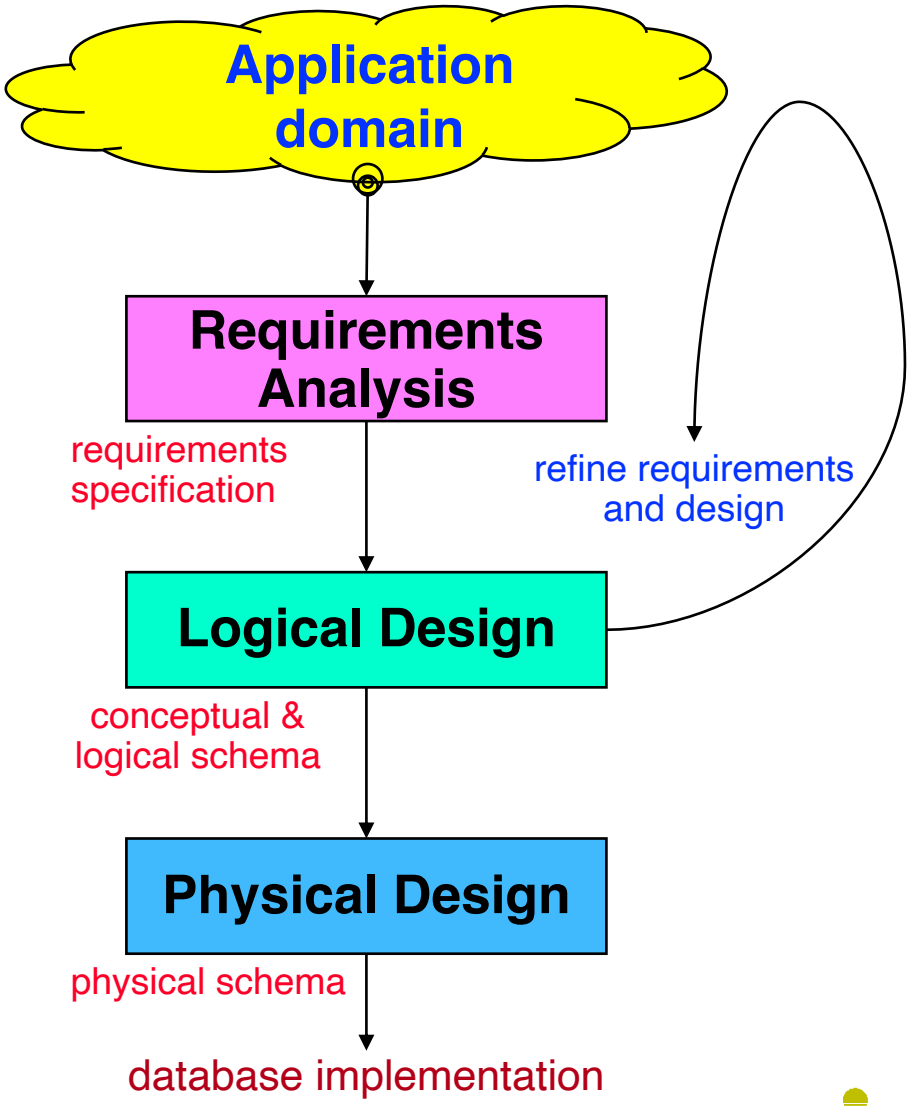
Entity-Relationship (E-R) Model — Data Structure Types
- Entity
- Attribute
- Entity Specialization/Generalization
- Relationship

Entity-Relationship (E-R) Model — Constraints
- Attribute — Domain, Key
- Entity Specialization/Generalization — Coverage
- Relationship — Cardinality, Participation, Exclusion

Analyzing Application Requirements / Making Design Choices

Reduction of E-R Schemas to Relational Schemas

DATABASE DESIGN PROCESS



Database Design Goals

1. Meet the data content requirements of users.
2. Provide a natural and easy-to-understand structuring of data.
3. Support data processing requirements and any performance objectives (e.g., response time, processing time, storage space, etc.).

E-R MODEL & DB DESIGN: OUTLINE

✓ Database Design Process

➡ **Entity-Relationship (E-R) Model — Data Structure Types**
- **Entity**
- **Attribute**
- **Entity Specialization/Generalization**
- **Relationship**

Entity-Relationship (E-R) Model — Constraints
- Attribute — Domain, Key
- Entity Specialization/Generalization — Coverage
- Relationship — Cardinality, Participation, Exclusion

Analyzing Application Requirements / Making Design Choices

Reduction of E-R Schemas to Relational Schemas

E-R MODEL & DATABASE DESIGN

EXERCISE 1

EXERCISE 1: UNIVERSITY APPLICATION

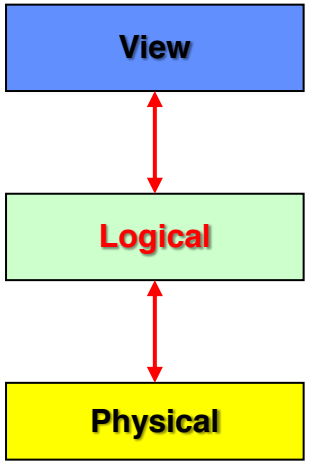
We want to record information about students, departments, courses and course teaching teams.

- For each student we store the student id, name and majors.
- For each department we store a unique code and name.
- For each course we store a unique course id, name, department and prerequisites.
- For each offering of a course, we store the section, semester and year.
- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course offering that a student takes we store the grade.
- Each course offering's teaching team has one or more staff, who is either an instructor or a TA.
- For each staff assigned to a course offering's teaching team we store the hkid, name, department and office number.
- For each instructor we store their academic title (e.g., professor).

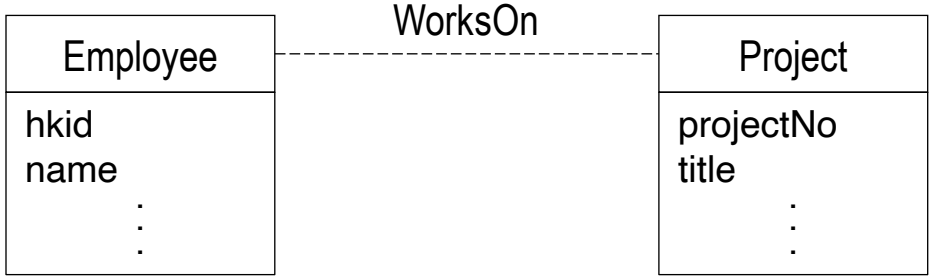
Construct an E-R diagram for the university application.

ENTITY-RELATIONSHIP (E-R) MODEL

The entity-relationship (E-R) model is used at the logical level to describe a database's overall structure.



- The E-R model employs three basic concepts to describe data.
  1. entity (something about which we want to keep data).
  2. attribute (properties of entities).
  3. relationship (among entities).



Why E-R model?
- expressiveness
- user communication
- DBMS independent

☞ **These are shown in an entity-relationship diagram (E-R diagram).**

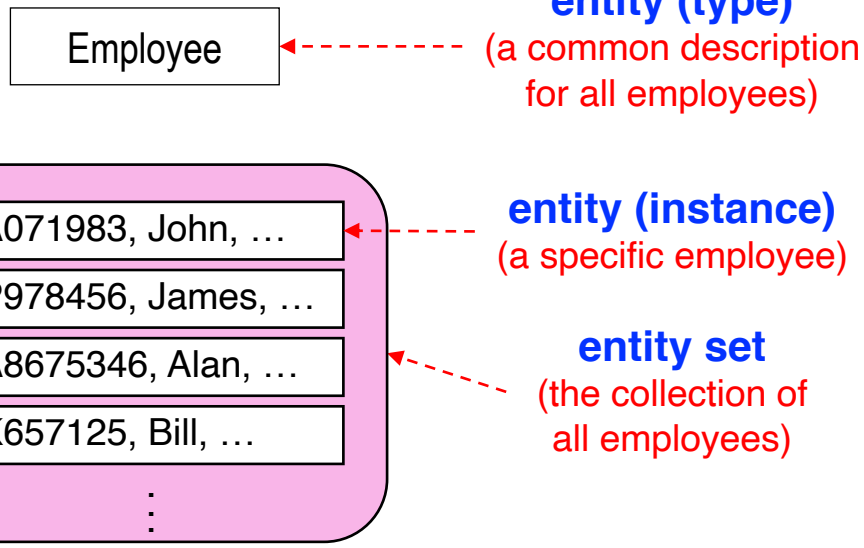
ENTITY

An *entity (type)* describes a set of entity instances with common:
– properties – relationships – semantics

Something we want to store data about in the application domain.

(E.g., employee, student, course, product, order, … .)

Notation:



An entity instance

- has identity.
  - It can be distinguished from other entity instances.
- represents some real-world thing.
  - It has meaning in the application domain.

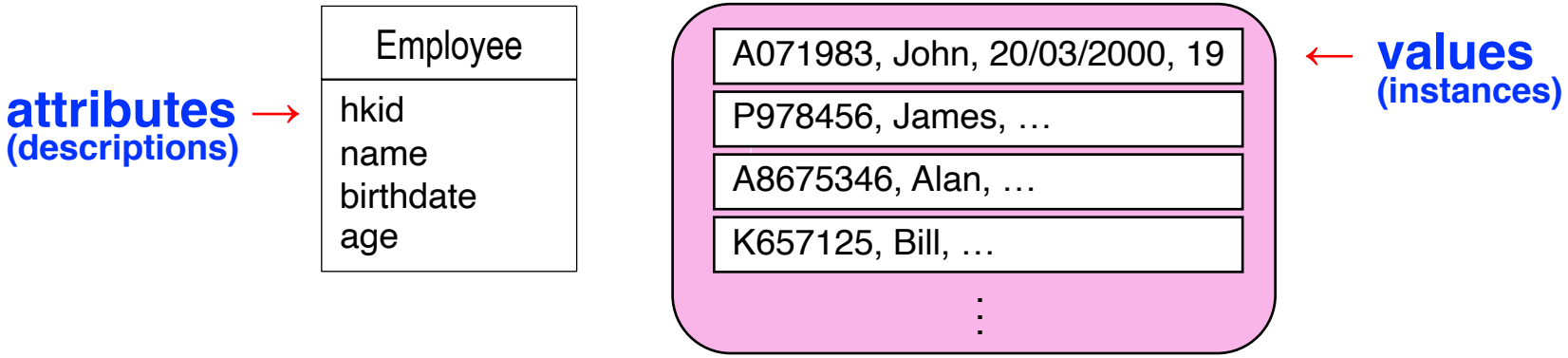
EXERCISE 1: UNIVERSITY APPLICATION—ENTITIES

- For each student we store the student id, name and majors.
- For each department we store a unique code and name.
- For each course we store a unique course id, name, department and prerequisites.
- For each offering of a course, we store the section, semester and year.
- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course offering that a student takes we store the grade.
- Each course offering's teaching team has one or more staff, who is either an instructor or a TA.
- For each staff assigned to a course offering's teaching team we store the hkid, name, department and office number.
- For each instructor we store their academic title (e.g., professor).

Student	Department	Course	Offering	Staff	Instructor	TA

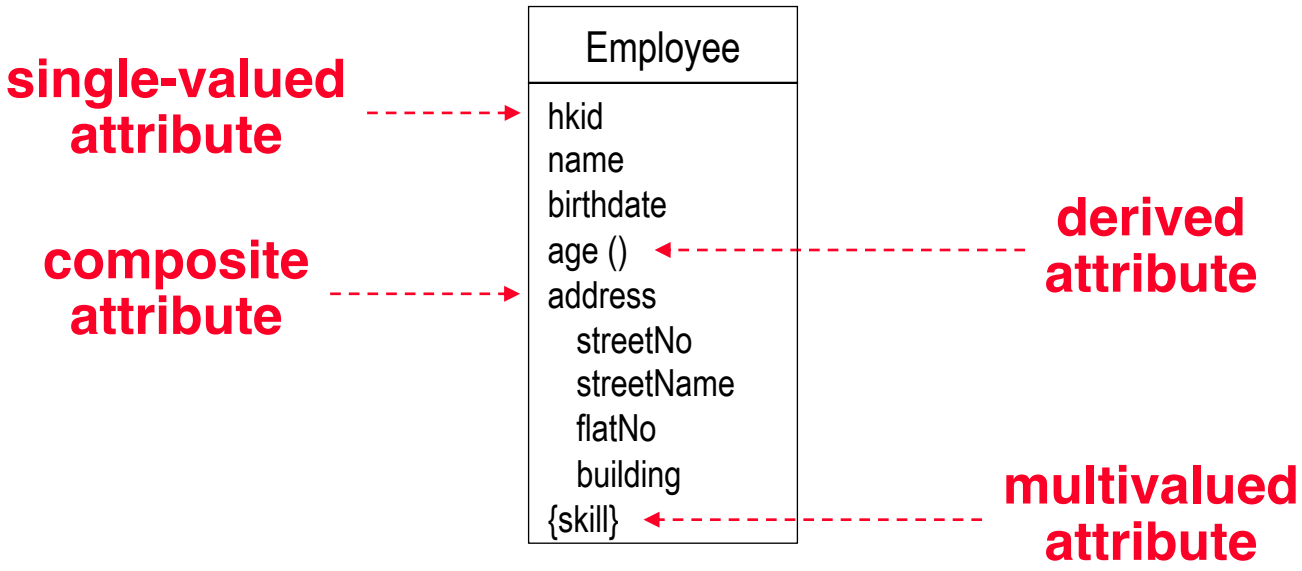
ATTRIBUTE

An *attribute* is a property of an entity and describes the data values of that property.



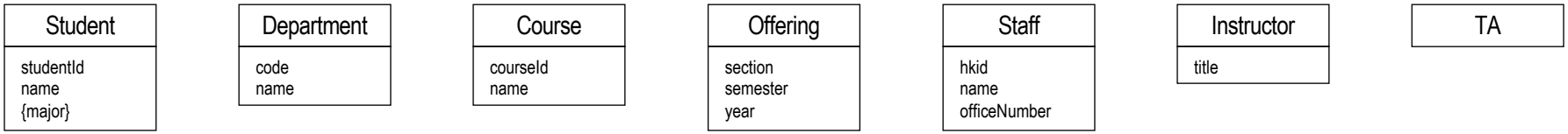
- Each attribute has a name that is unique within an entity (but not across entities).
- Most attribute values are physically stored (base attribute); some may be calculated using stored values (derived attribute).
- An attribute value may be null (missing, unknown, not applicable).

TYPES OF ATTRIBUTES AND NOTATION



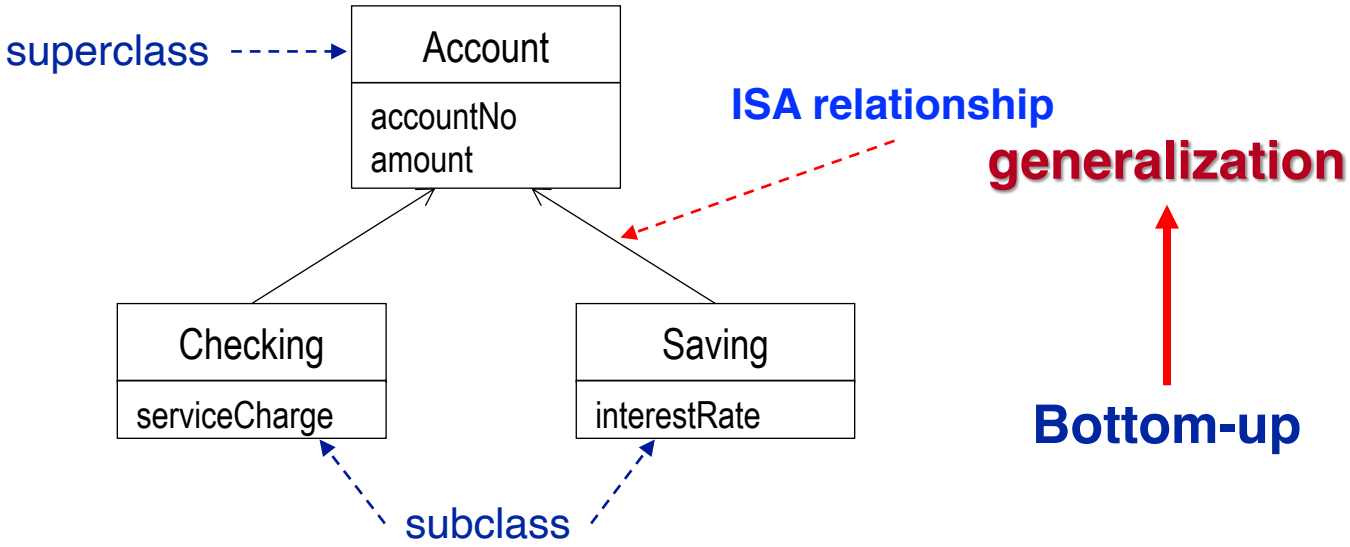
EXERCISE 1: UNIVERSITY APPLICATION—ENTITY ATTRIBUTES

- For each student we store the student id, name and majors.
- For each department we store a unique code and name.
- For each course we store a unique course id, name, department and prerequisites.
- For each offering of a course, we store the section, semester and year.
- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course offering that a student takes we store the grade.
- Each course offering's teaching team has one or more staff, who is either an instructor or a TA.
- For each staff assigned to a course offering's teaching team we store the hkid, name, department and office number.
- For each instructor we store their academic title (e.g., professor).



ENTITY GENERALIZATION/SPECIALIZATION

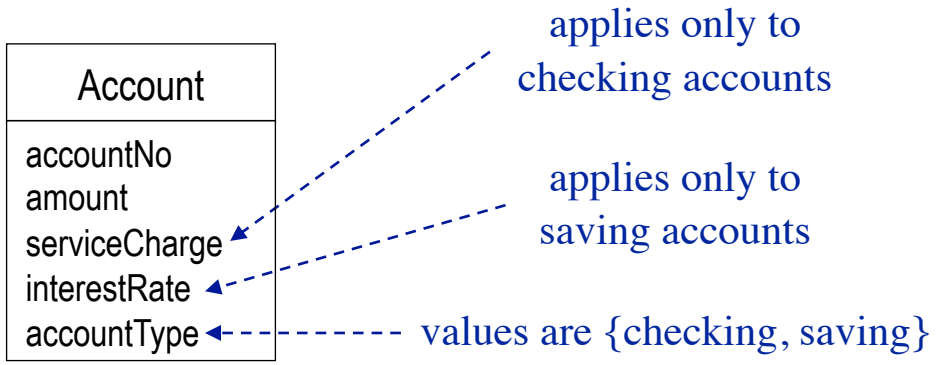
Generalization/specialization **is a relationship between the same kind of entities playing different roles.**



☞ In this example, subclass membership is user-defined (i.e., determined by the schema designer and not based on any attribute).

ENTITY GENERALIZATION/SPECIALIZATION (cont'd)

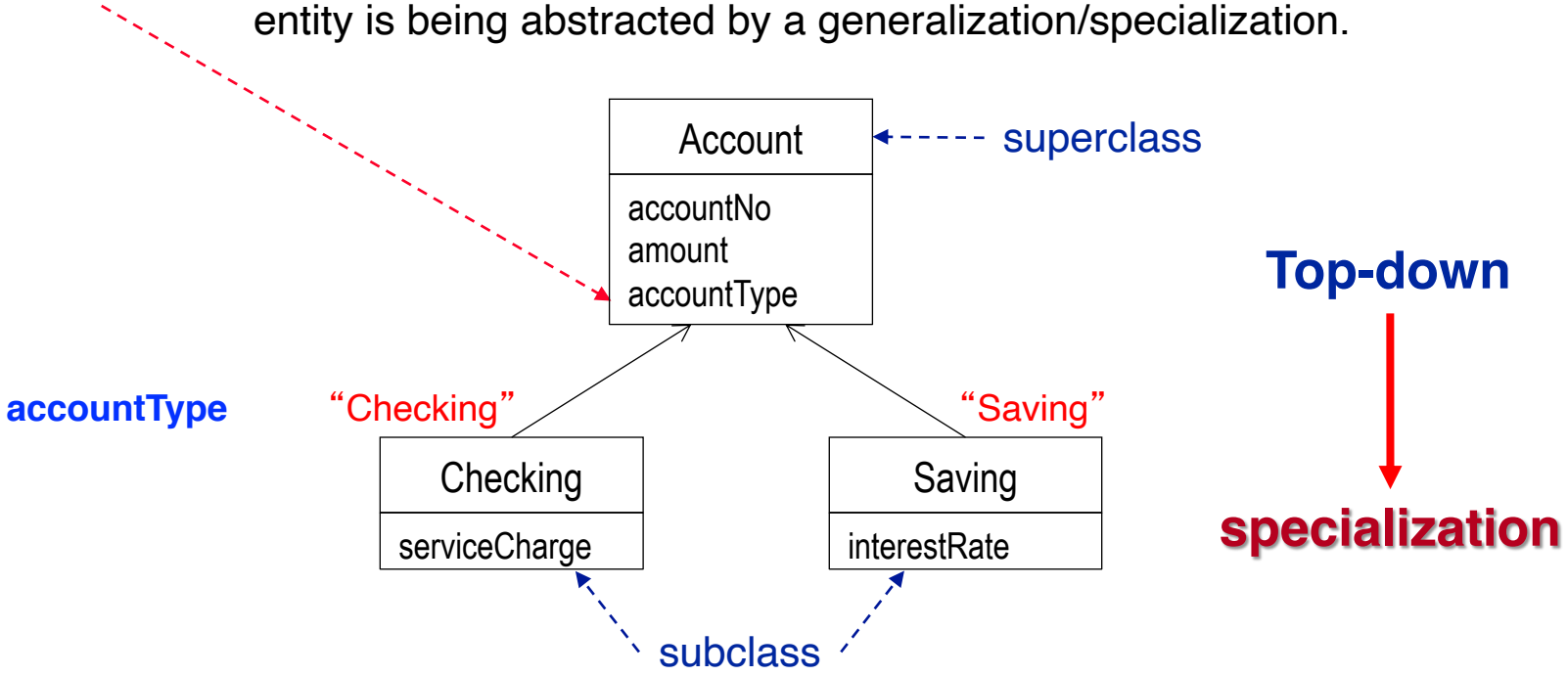
Can also be applied top-down (attribute-defined).



ENTITY GENERALIZATION/SPECIALIZATION (cont'd)

Can also be applied top-down (attribute-defined).

discriminator: An attribute of enumeration type that indicates which property of an entity is being abstracted by a generalization/specialization.



☞ In this example, subclass membership is determined by a predicate on an attribute (i.e., the discriminator attribute) of the superclass.

INHERITANCE

***Inheritance* is the taking up of properties by a subclass from its superclass.**

- We extract the *common* attributes and relationships, associate them with the superclass and inherit them to the subclass(es).
  - ✓ Reduces redundancy of descriptions.
  - ✓ Promotes reusability of descriptions.
  - ✓ Simplifies modification of descriptions.

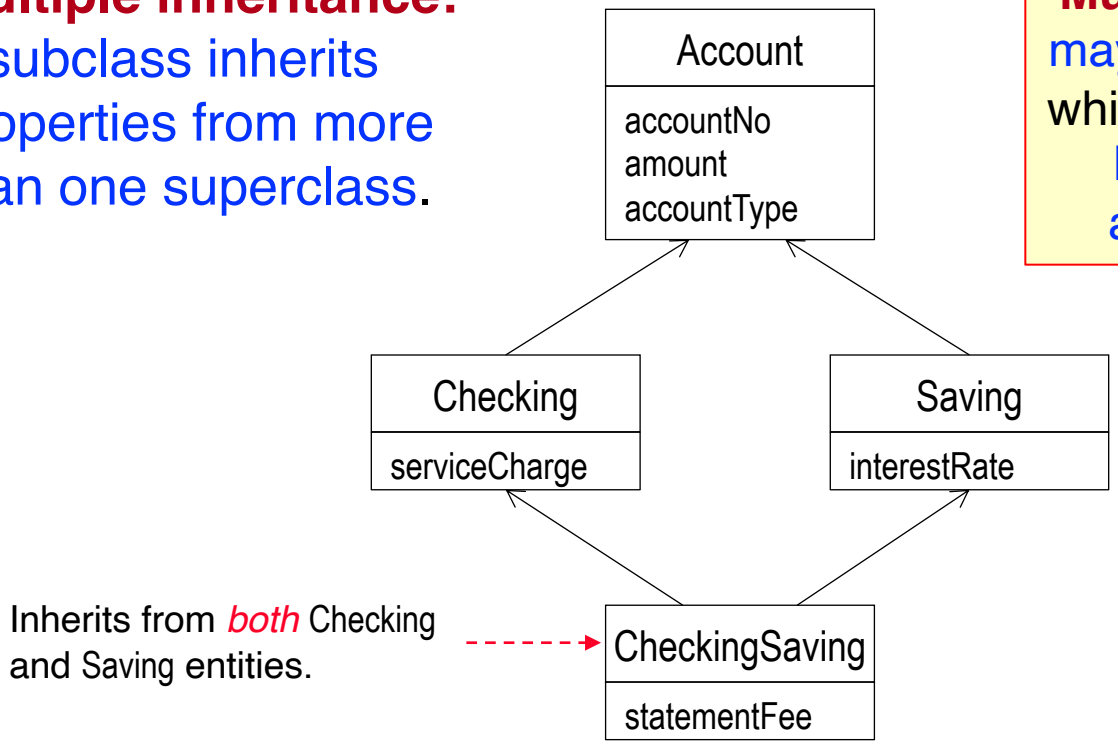
  ☞ **We only define an entity's properties in one place.**

- A subclass may add new properties (attributes, relationships).

Design Guideline: Inheritance should not exceed 2-3 levels.

SINGLE VS. MULTIPLE INHERITANCE

Multiple inheritance:
a subclass inherits properties from more than one superclass.



Multiple inheritance may result in conflicts, which can be resolved by redefining an attribute's name.

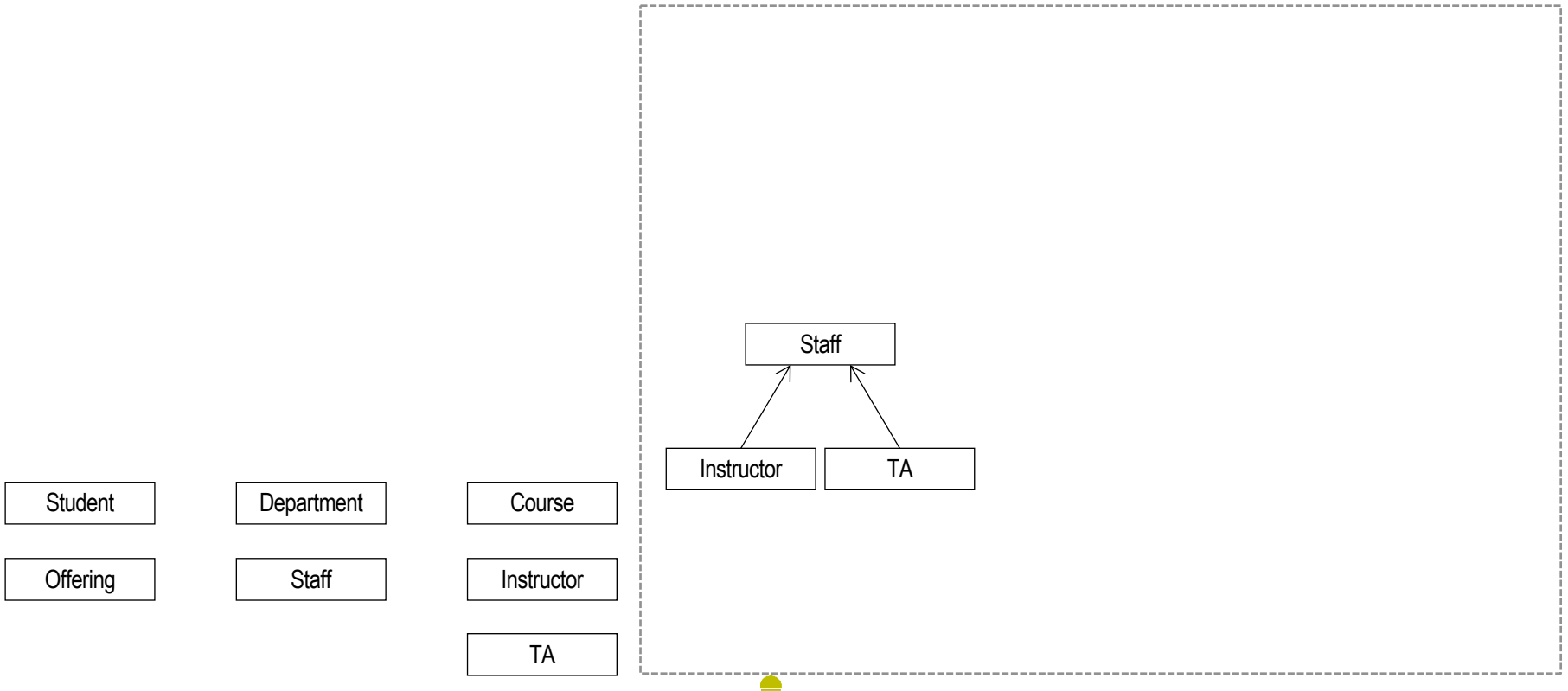
For multiple inheritance, a property *from the same ancestor entity* found along more than one path is *inherited only once*.

EXERCISE 1: UNIVERSITY APPLICATION—ENTITY GENERALIZATION

- Each course offering's teaching team has one or more staff, who is either an instructor or a TA.

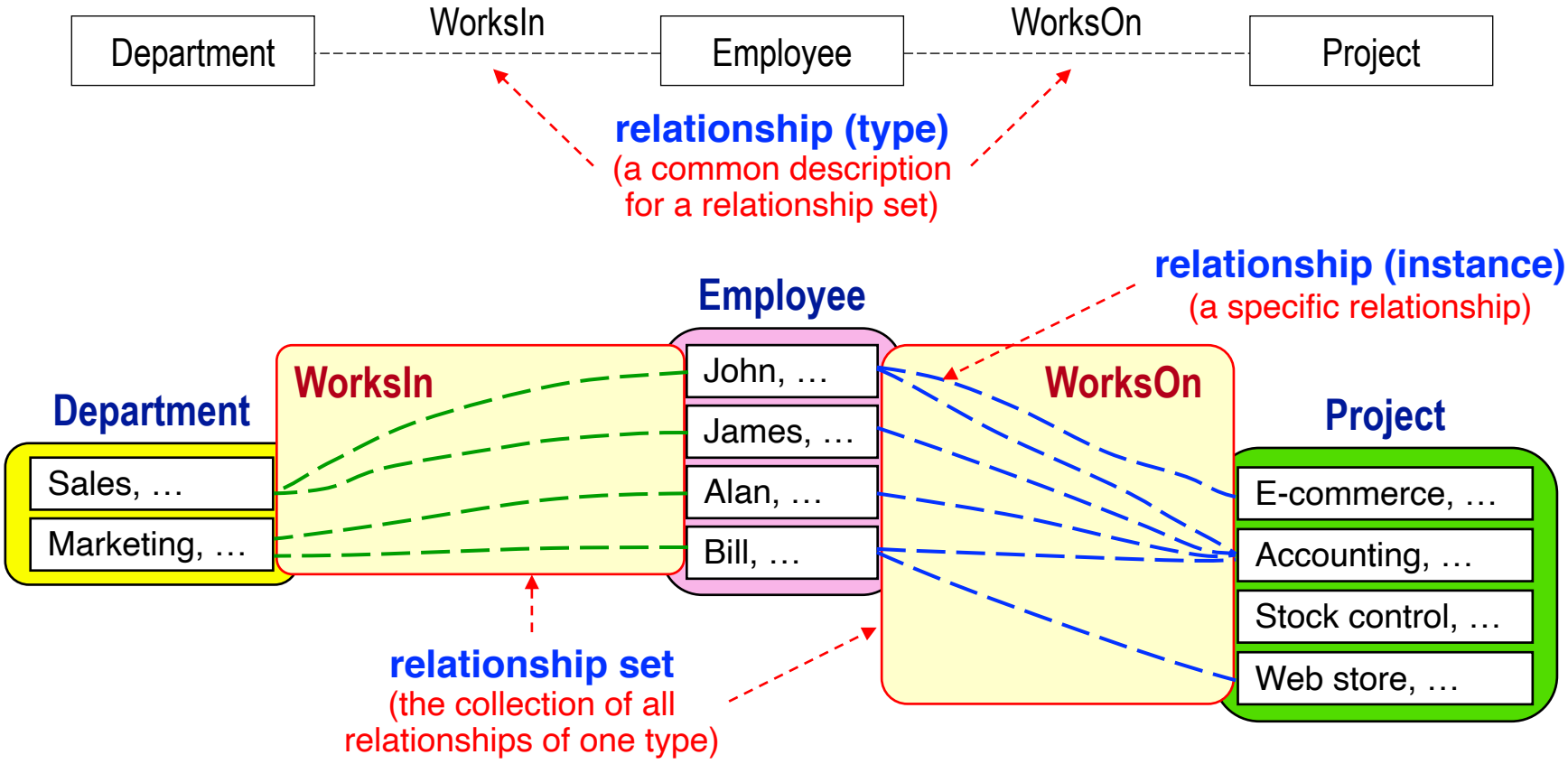
What should be the generalization?

⟹ Staff superclass; Instructor, TA subclasses.



RELATIONSHIP

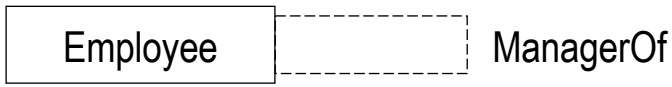
A *relationship (type)* is a description of a set of relationships with common properties and semantics.



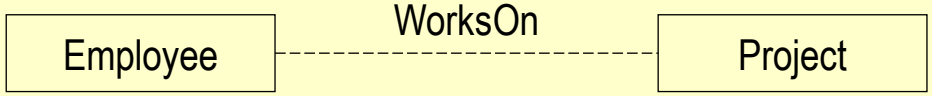
RELATIONSHIP DEGREE

- The number of entities that participate in a relationship.

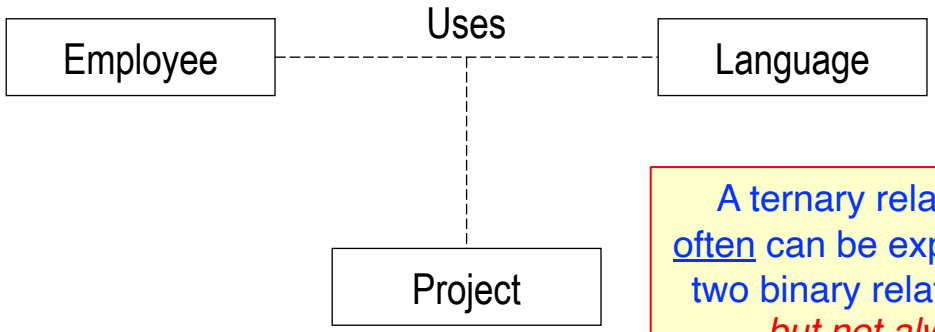
unary (reflexive) – relates *the same* entity (to itself)



binary – relates *two* different entities



ternary – relates *three* different entities



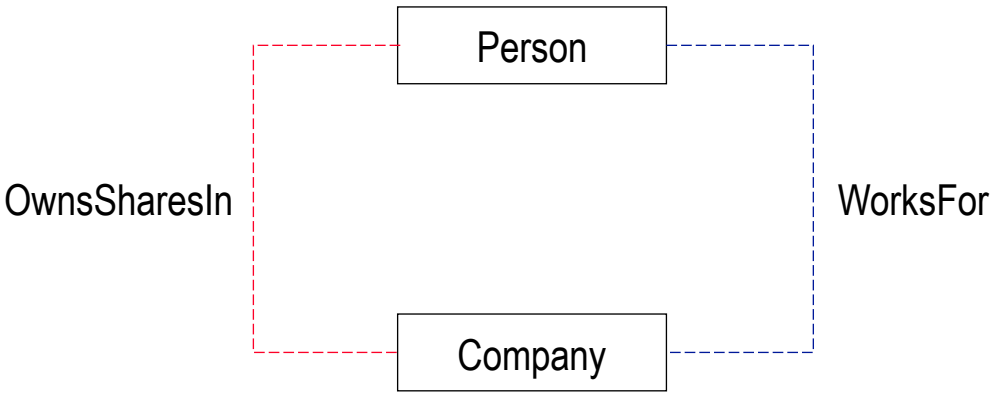
Higher degrees are extremely rare!

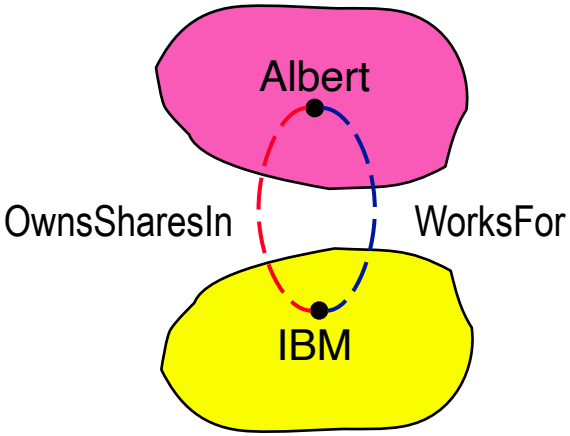
A ternary relationship often can be expressed as two binary relationships, *but not always*.

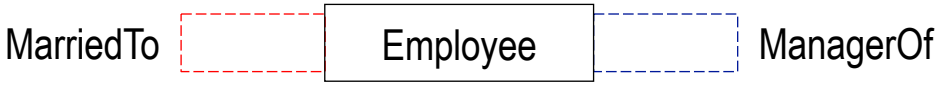
In practice, the vast majority of relationships are binary.
(We will use only unary or binary relationships in this course.)

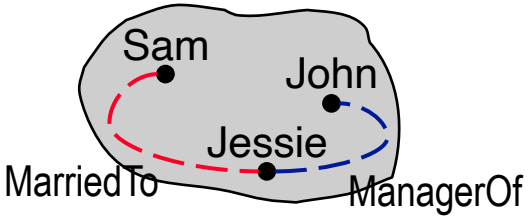
RELATIONSHIP EXAMPLES

There can be several relationships between entities.







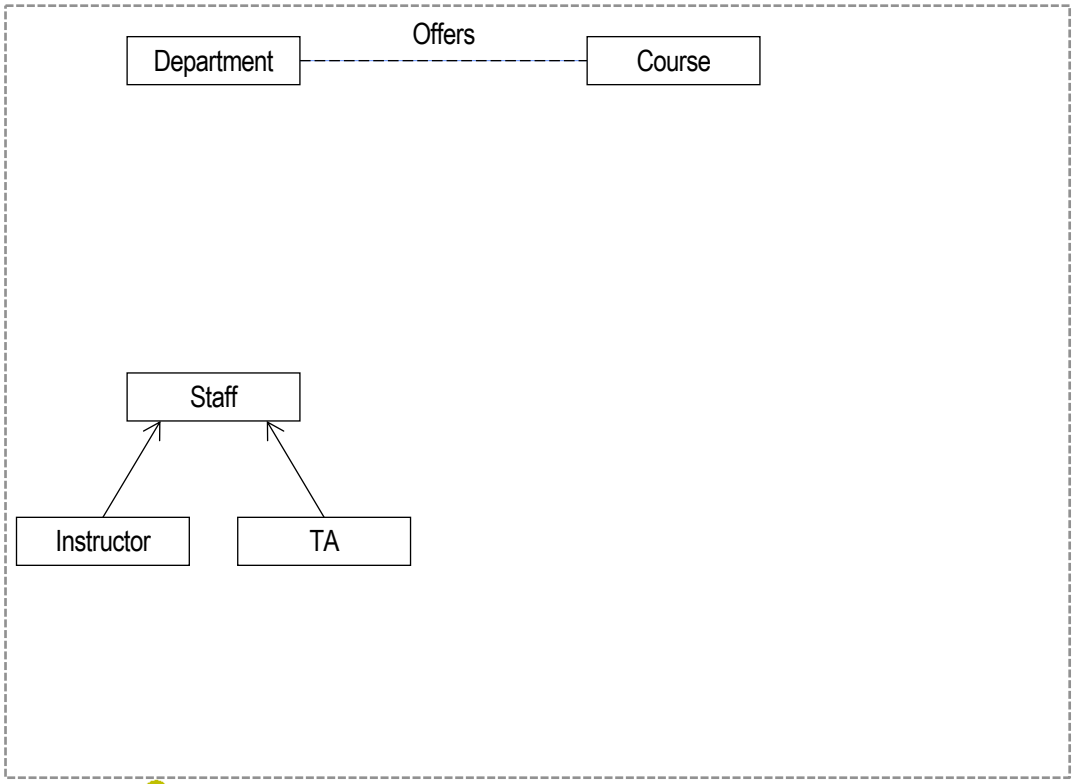


EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- For each course we store a unique course id, name, department and prerequisites.

What should be related?

⟹ Course related to Department.



Student

Department

Course

Offering

Staff

Instructor

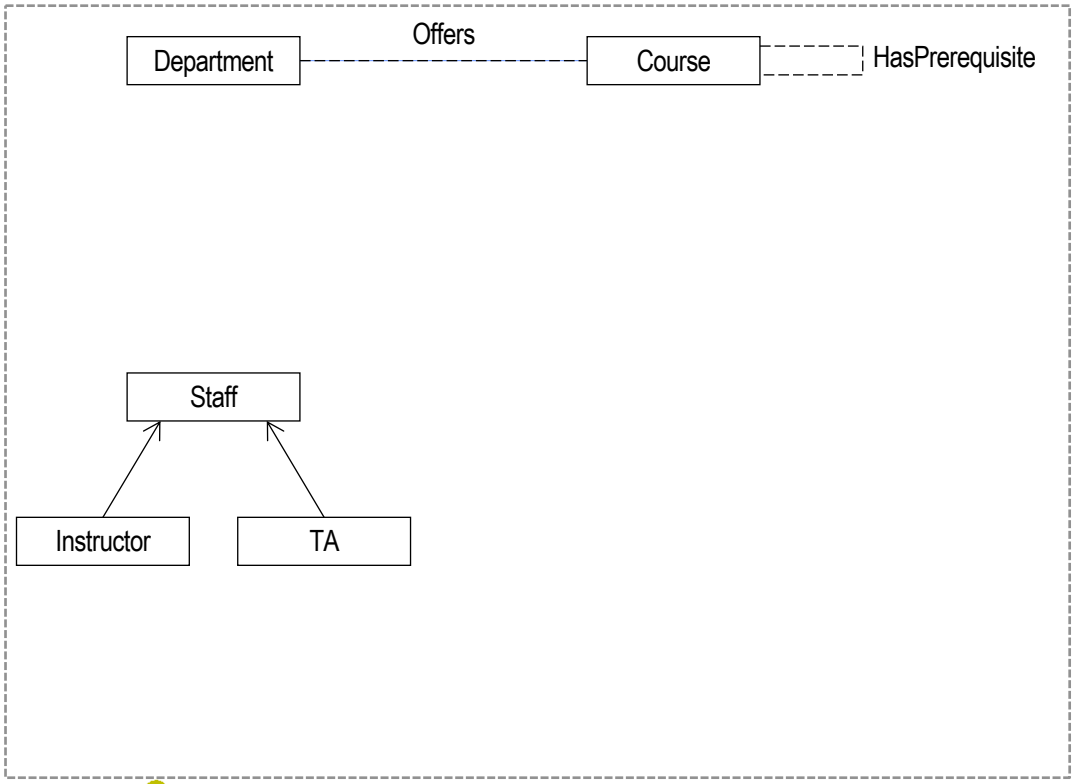
TA

EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- For each course we store a unique course id, name, department and prerequisites.

What should be related?

⟹ Course related to Course (unary relationship).



Student | Department | Course

Offering | Staff | Instructor

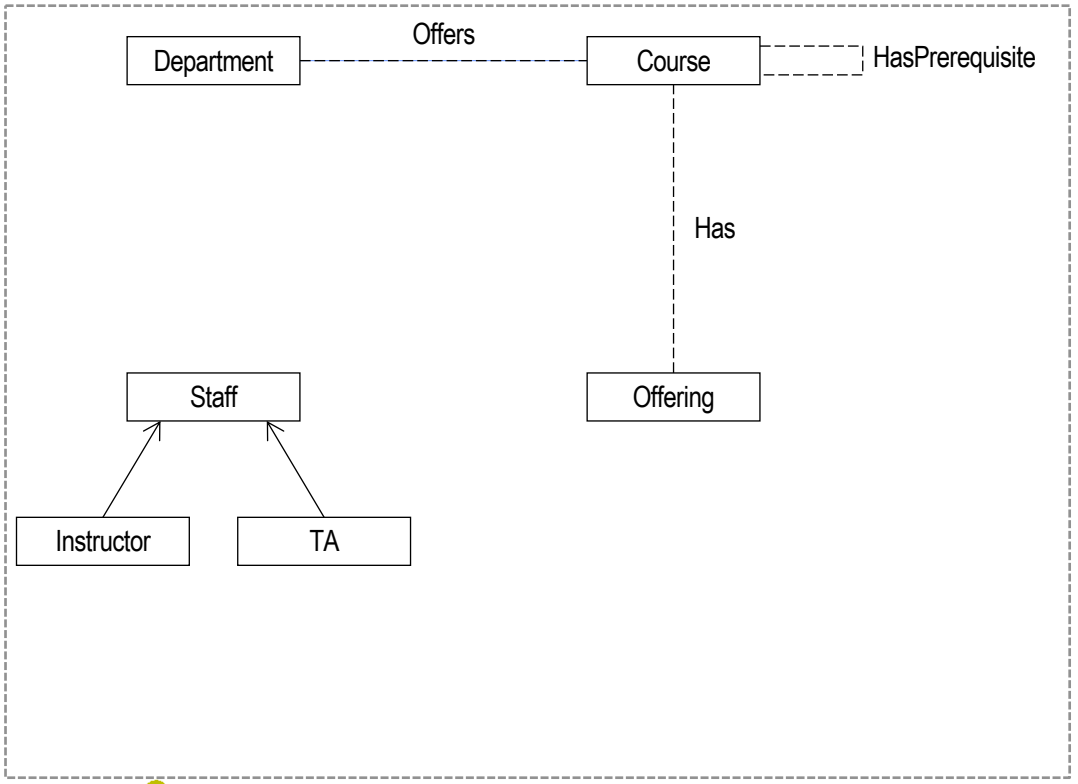
TA

EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- For each offering of a course, we store the section, semester and year.

What should be related?

⟹ Offering related to Course.



Student | Department | Course

Offering | Staff | Instructor

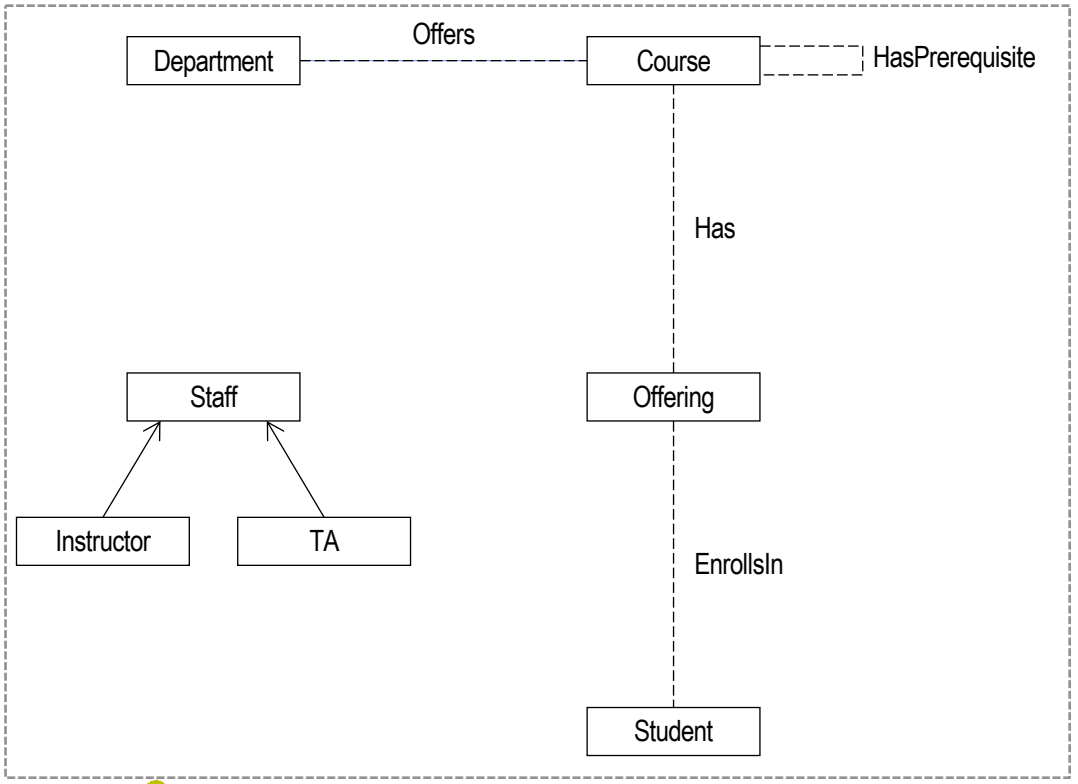
TA

EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course that a student takes we store the grade.

What should be related?

⟹ Student related to Offering.



Student | Department | Course

Offering | Staff | Instructor

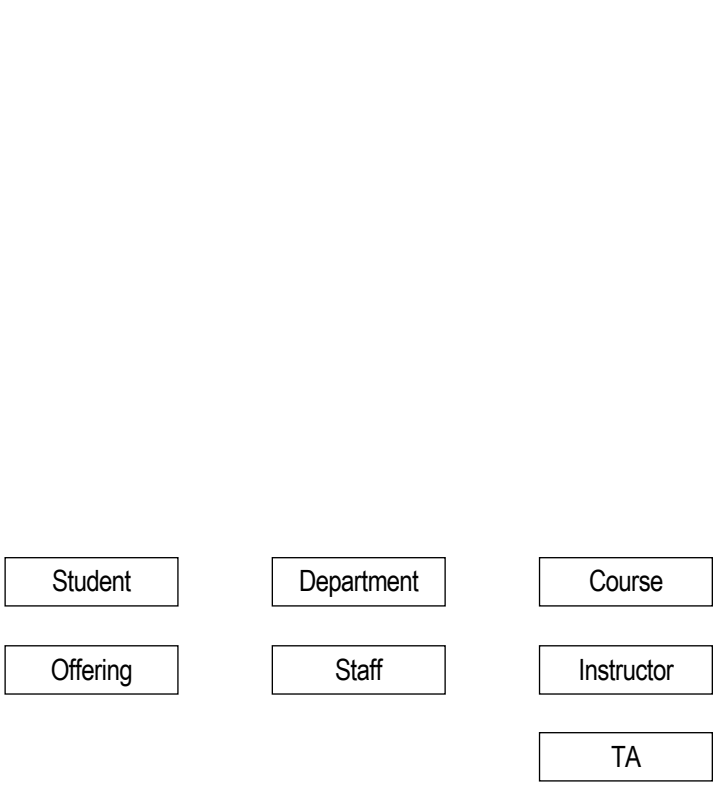
TA

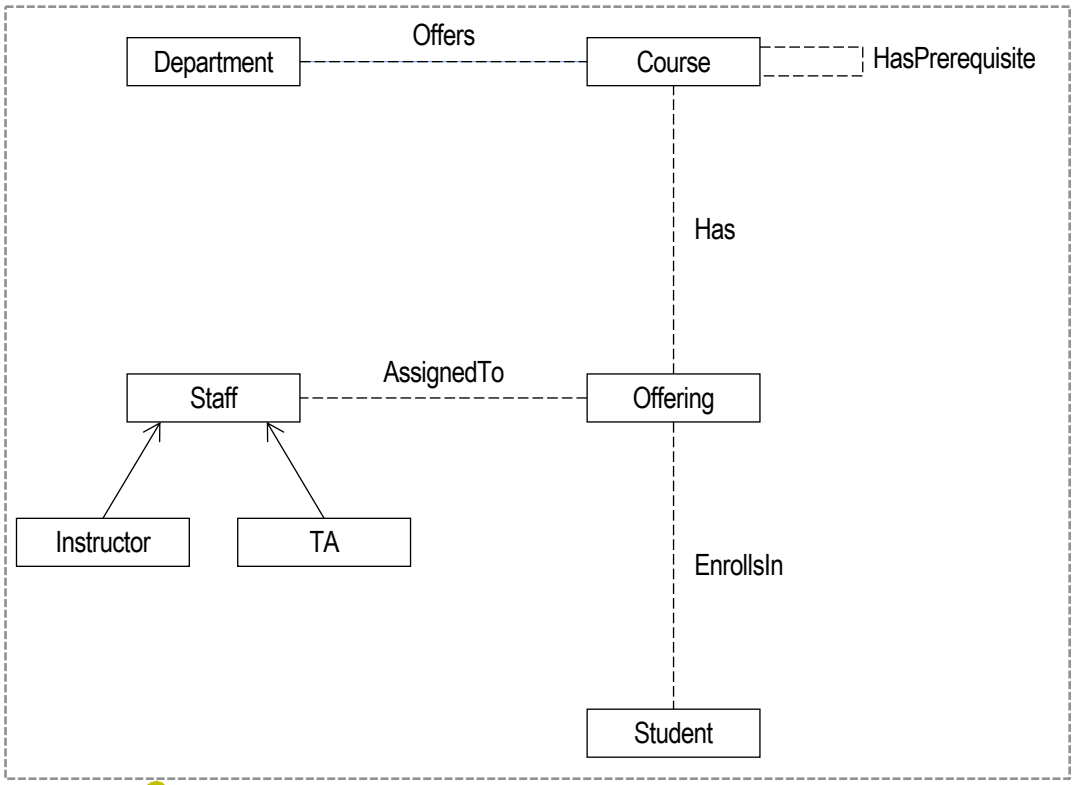
EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- For each staff assigned to a course offering's teaching team we store the hkid, name, department and office number.

What should be related?

⟹ Staff related to Offering.



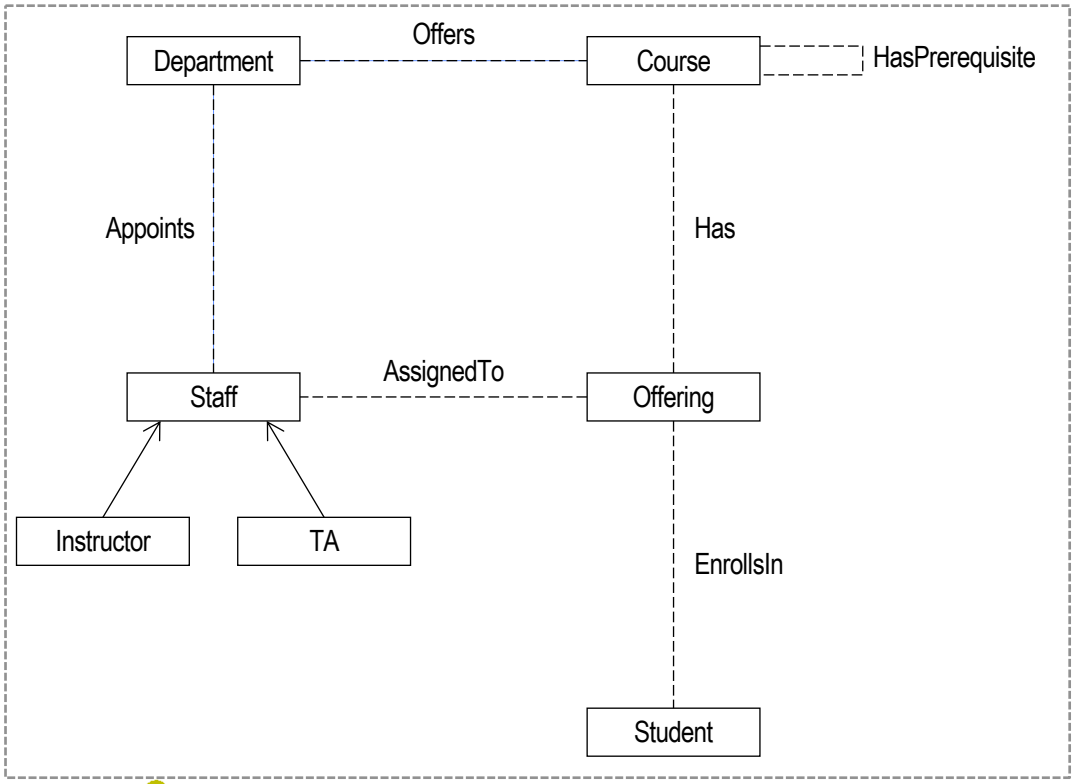


EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIPS

- For each staff assigned to a course offering's teaching team we store the hkid, name, department and office number.

What should be related?

⟹ Staff related to Department.



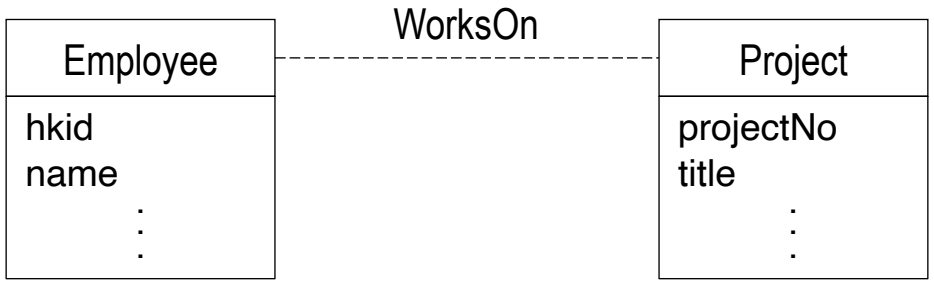
Student | Department | Course

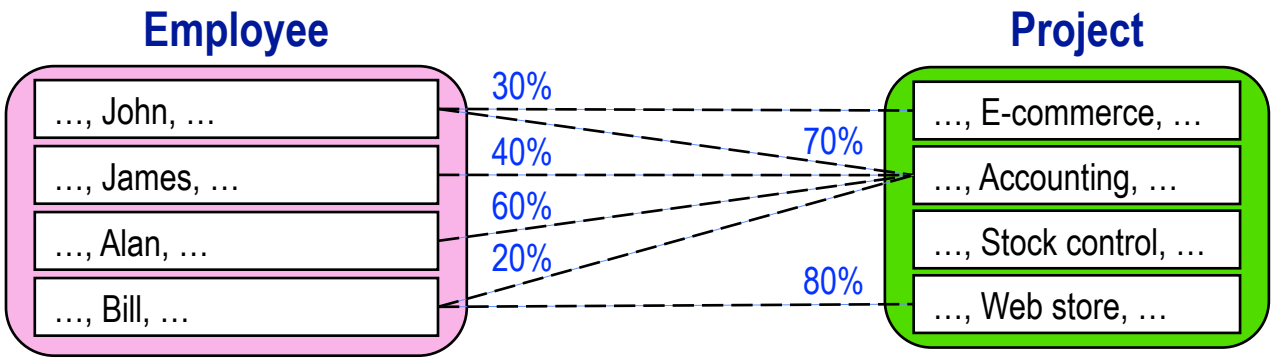
Offering | Staff | Instructor

TA

RELATIONSHIP ATTRIBUTES

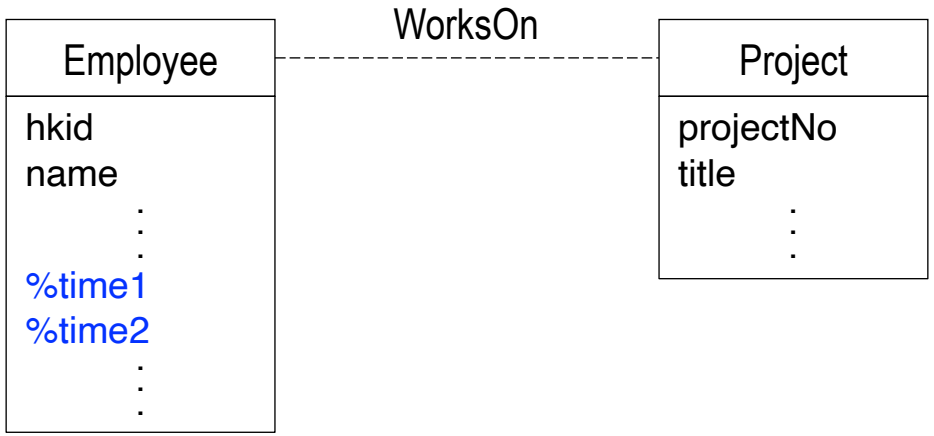
- We want to represent the percentage time worked on a project.

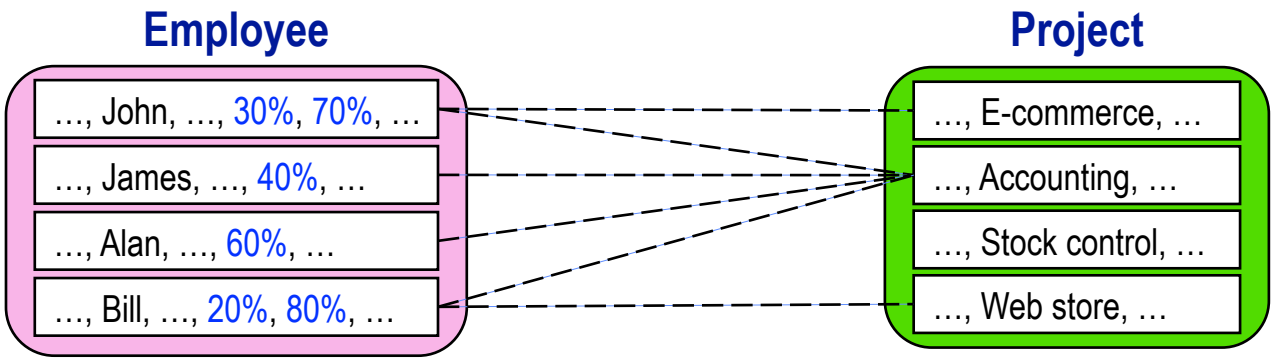




RELATIONSHIP ATTRIBUTES (cont'd)

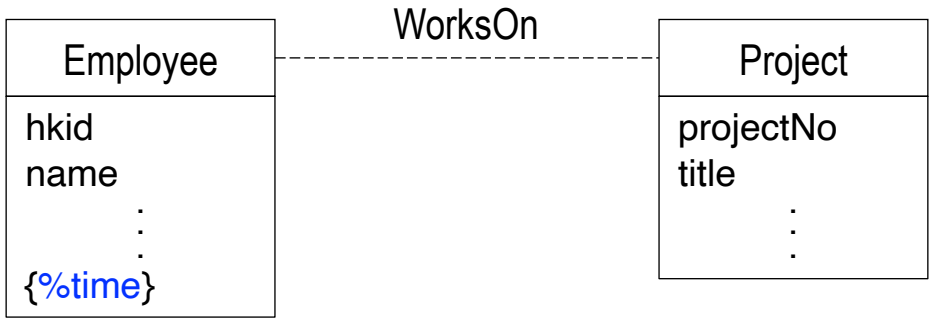
Option 1: Use many attributes (e.g., in Employee). **Is this OK?**

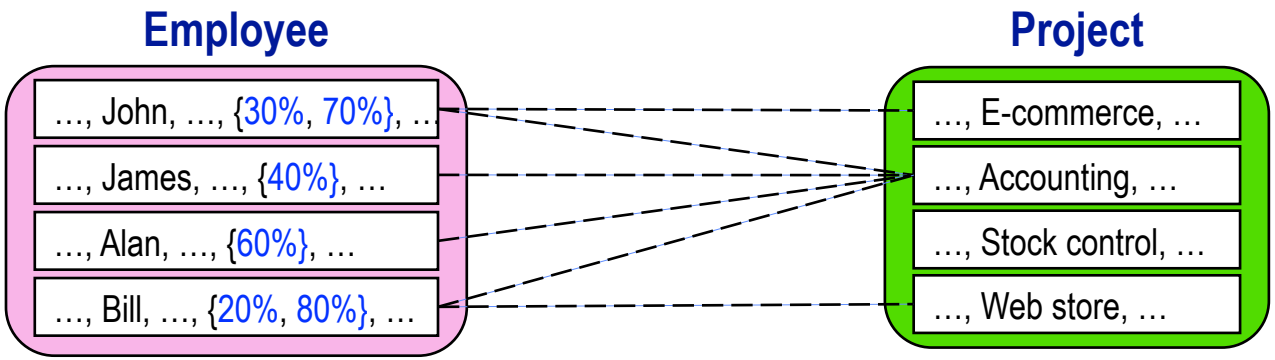




RELATIONSHIP ATTRIBUTES (cont'd)

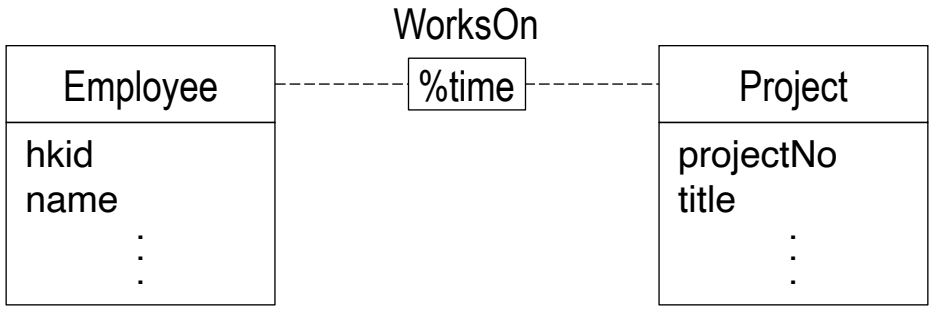
Option 2: Use a multivalued attribute (e.g., in Employee). **Is this OK?**

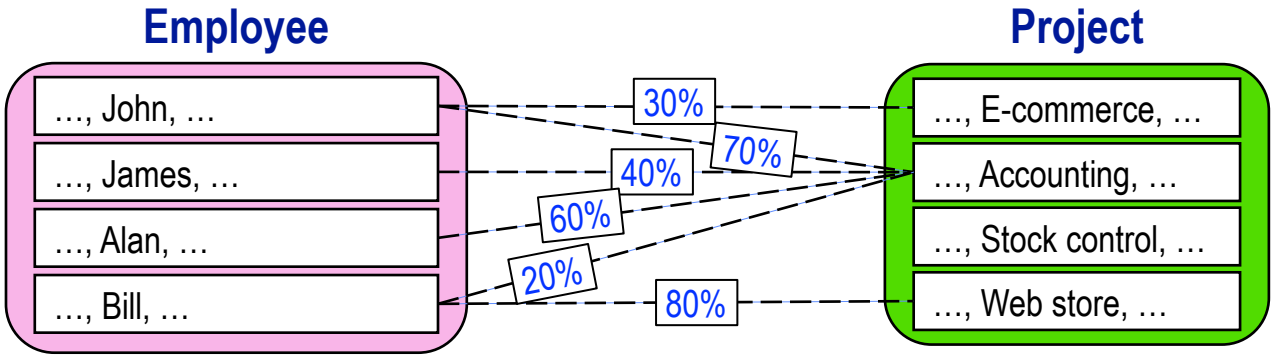




RELATIONSHIP ATTRIBUTES (cont'd)

Option 3: Allow relationships to have attributes. **Is this OK?**





EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP ATTRIBUTES

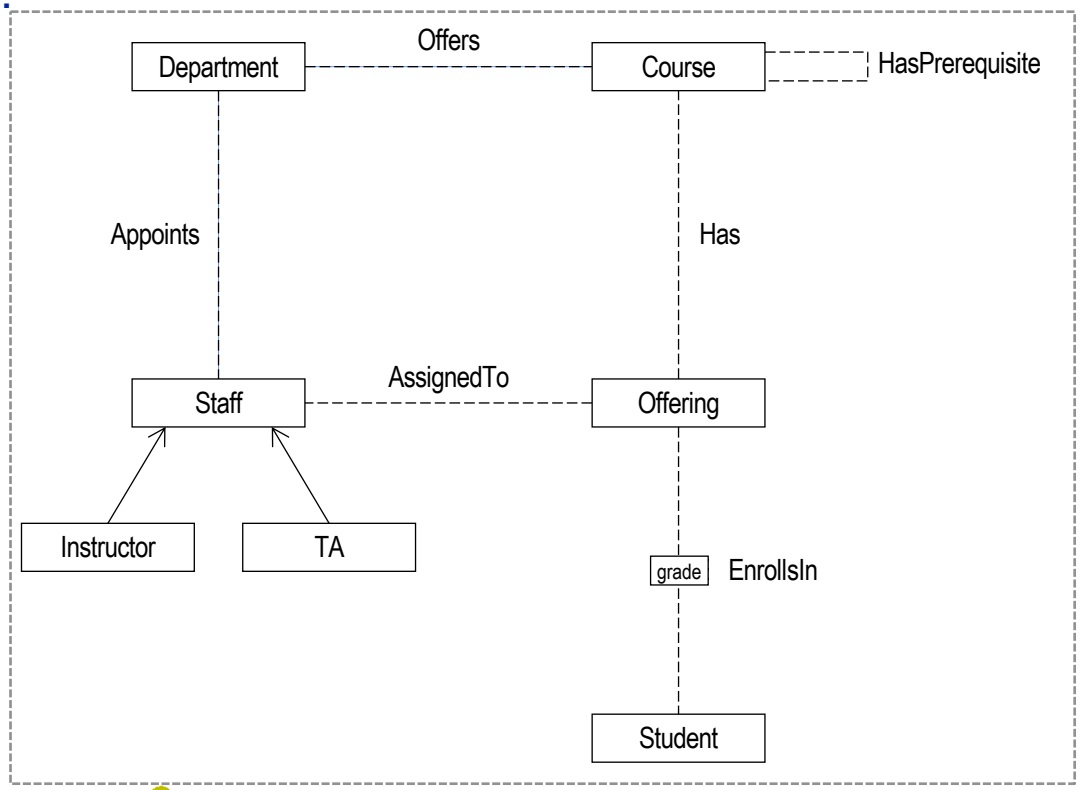
- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.
- For each course that a student takes we store the grade.

Any relationship attributes?

⟹ Yes — grade attribute

Add where?

⟹ To EnrollsIn relationship.



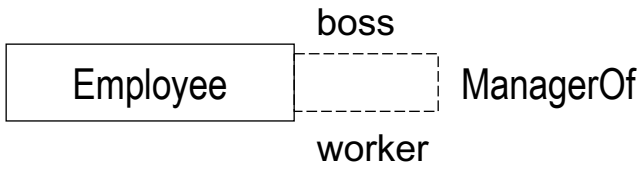
Student | Department | Course

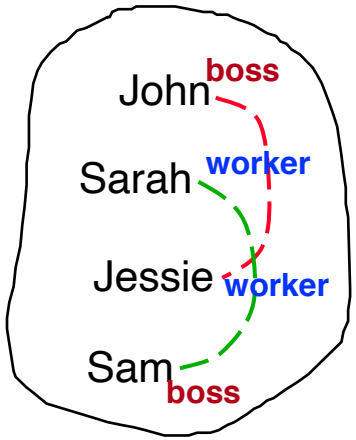
Offering | Staff | Instructor

TA

RELATIONSHIP ROLE NAMES

A *role name* is assigned to one end of a relationship to identify the role that the entity at that end plays in the relationship.





Who is the boss and who is the worker?

A role name disambiguates the role that an entity plays in a relationship.

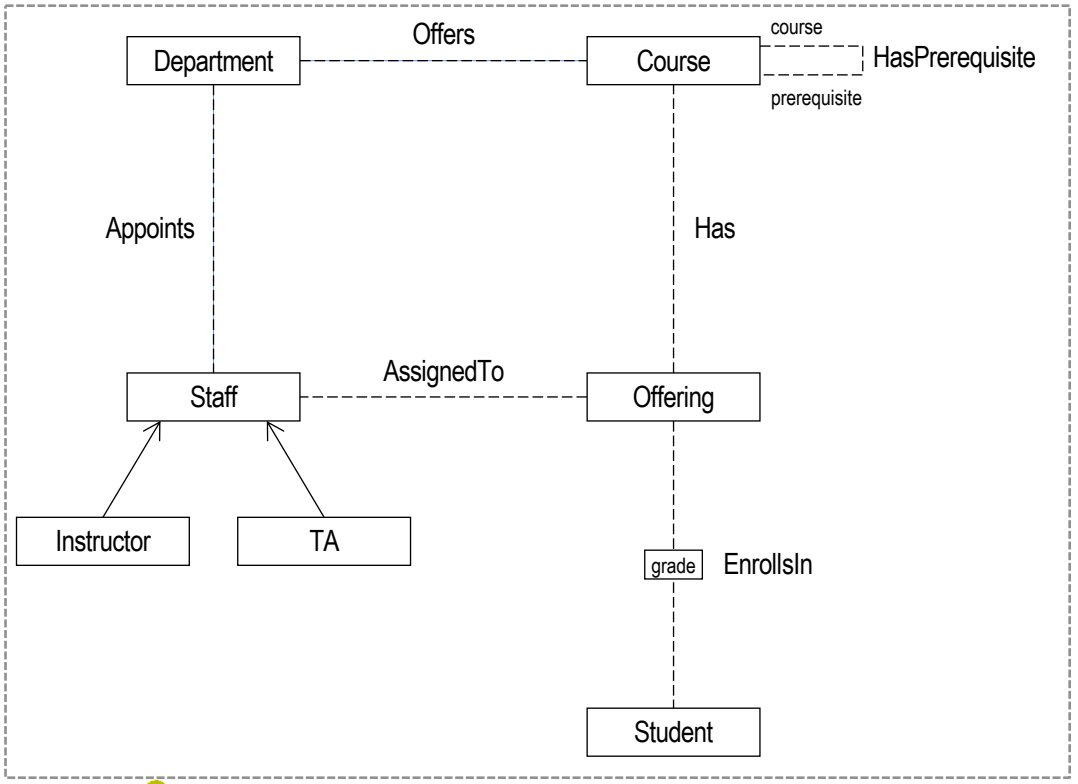
It is necessary to use role names for unary relationships (i.e., when a relationship relates instances from the same entity).

EXERCISE 1: UNIVERSITY APPLICATION— RELATIONSHIP ROLE NAMES

- For each course we store a unique course id, name, department and prerequisites.

Any role names?

⟹ Yes — add role names to HasPrerequisite.

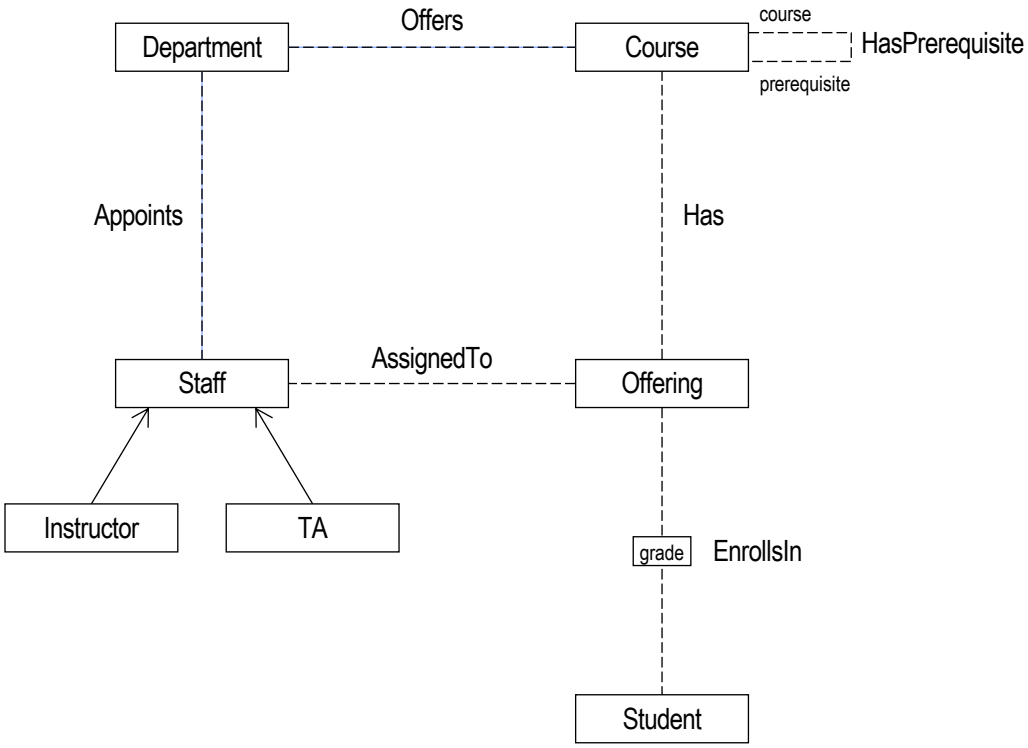


Student | Department | Course

Offering | Staff | Instructor

TA

EXERCISE 1: UNIVERSITY APPLICATION— E-R DIAGRAM



Student	Department	Course	Offering	Staff	Instructor	TA
studentId name {major}	code name	courseId name	section semester year	hkid name officeNumber	title	

EXERCISE 2: BUS COMPANY

We want to keep track of bus routes and schedules for a bus company.

- Each bus route has a unique route number, a departure station and a destination station.
- For each bus route, there is a schedule, which records all the departure times of buses.
- For each departure time of each route, a driver and a bus can be assigned; however, information about the driver or the bus may sometimes be missing.
- A driver has a unique employee id, a name and a phone number.
- A bus is identified by its license number and has a maximum seating capacity.

Construct an E-R diagram for the bus company application.

What is a Schedule?

Route 1		
Departure time	Driver	Bus
11:00	Bill	1
12:00	Sarah	2
13:00	Bill	5

Route 2		
Departure time	Driver	Bus
9:00	Al	3
11:00	Cindy	4
13:00	Al	3
15:00	Mark	5

Route 3		
Departure time	Driver	Bus
9:00	John	6
15:00	Sarah	2