# DSAA 5012
# Advanced Database Management for Data Science

## LECTURE 3
## ENTITY-RELATIONSHIP (E-R) MODEL AND DATABASE DESIGN

# E-R MODEL & DB DESIGN: OUTLINE

✓ Database Design Process

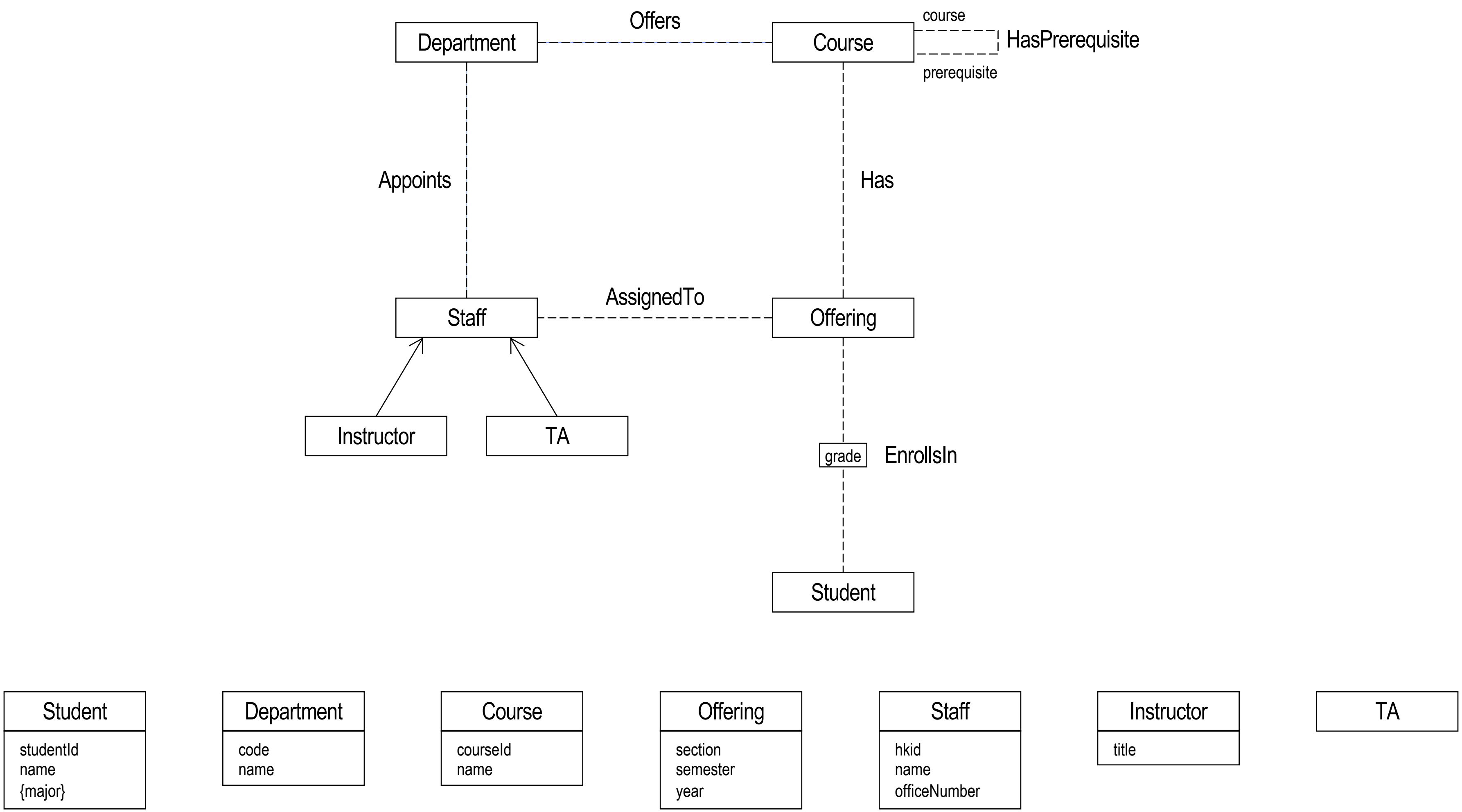
✓ Entity-Relationship (E-R) Model — Data Structure Types
- Entity
- Attribute
- Entity Generalization/Specialization
- Relationship

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

Analyzing Application Requirements / Making Design Choices

Reduction of E-R Schemas to Relational Schemas

# EXERCISE 1: UNIVERSITY APPLICATION— E-R DIAGRAM



# E-R MODEL: CONSTRAINTS

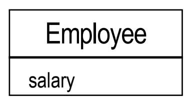
A constraint is a logical restriction or property of data that for any set of data values:

- we can determine whether the constraint is true or false;
- we expect the constraint to be always true;
- we can enforce the constraint.

## Examples:

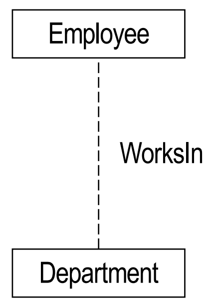
**on attributes**

salary is between $0 and $100,000

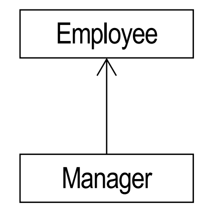



**on relationships**

every employee works in *at most* one department




not every employee is a manager

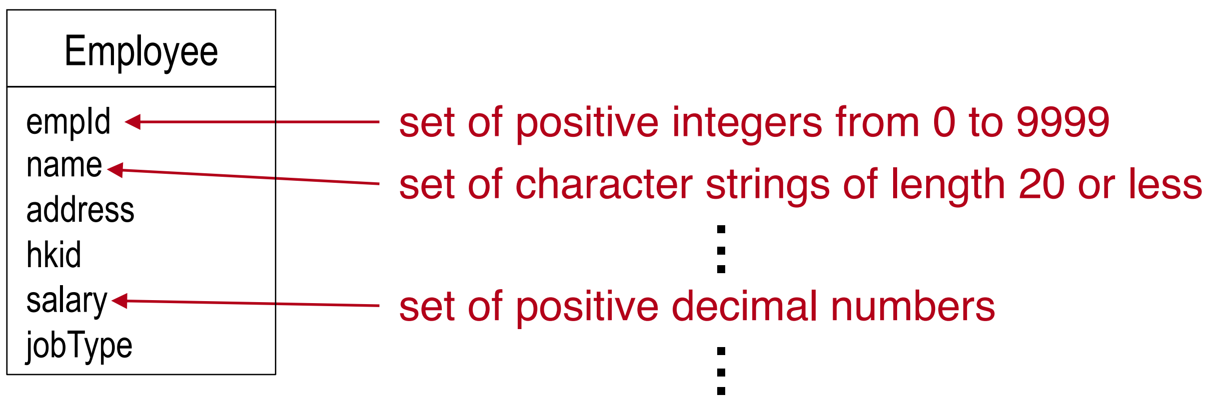



☞ **Constraints add additional semantics (meaning) to data**

**(so as to more accurately reflect the application requirements).**

# ATTRIBUTE CONSTRAINTS: DOMAIN

**A *domain constraint* restricts an attribute to have only certain values.**




A domain constraint can be specified as a type for the attribute and/or a logical predicate that restricts the values.

# ATTRIBUTE CONSTRAINTS: KEY

| Employee |
| --- |
| empId |
| name |
| address |
| hkid |
| salary |
| jobType |

- If the values of some attributes uniquely identify an entity instance, then they are a key for the entity.

- A candidate key is a minimal set of attributes (i.e., all attributes are needed) that uniquely identifies an entity instance.

  ☞ An entity may have more than one candidate key.

- One candidate key is selected by the database designer to be the primary key.

  ☞ **This has enforcement implications for implementation.**

  **primary key** ⟹ uniqueness is automatically enforced by a DBMS

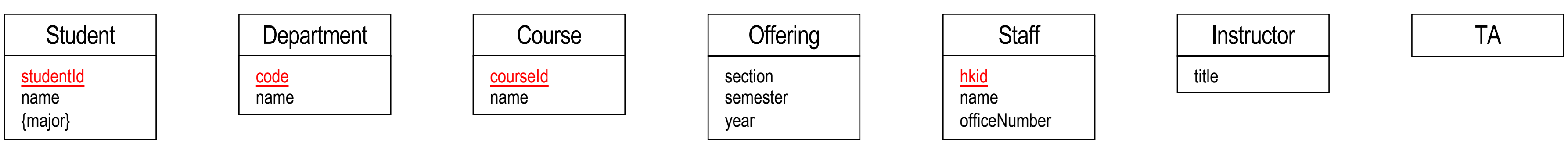
  **other candidate keys** ⟹ uniqueness is not automatically enforced by a DBMS

- A candidate/primary key can be composed of a set of attributes ⟹ composite key.

| EnrollsIn |
| --- |
| studentId |
| courseCode |
| grade |

# EXERCISE 1: UNIVERSITY APPLICATION— KEYS OF ENTITY TYPES

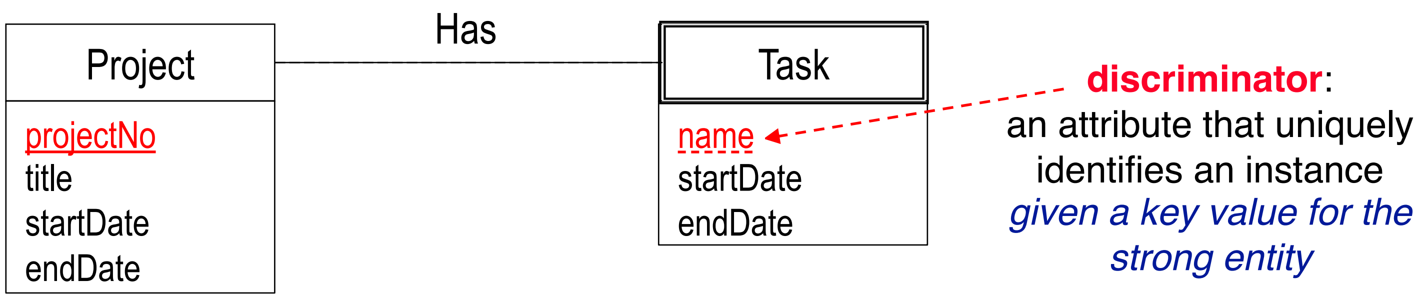
- For each student we store the student id, name and majors.
- For each department we store a unique code and name.
- For each course we store a unique course id, name, department and prerequisites.
- For each offering of a course, we store the section, semester and year.
- Each student must enroll in 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).



# STRONG ENTITY VS. WEAK ENTITY: KEY

**Strong entity**: An entity that has a primary key.

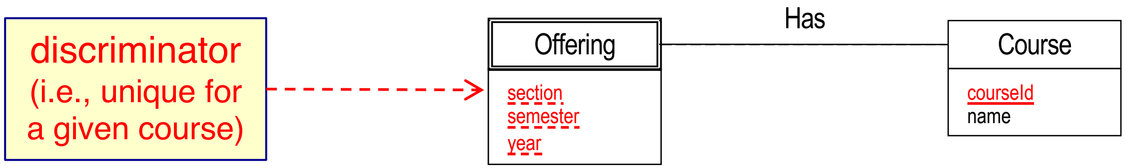
**Weak entity**: An entity that does not have a primary key.




- A weak entity must be associated with a strong entity, called the identifying entity, to be meaningful.
  - ☞ A weak entity depends on its identifying entity for its existence.
- The relationship associating the weak entity to the strong entity is called the identifying relationship (shown as a solid line).
- A discriminator, *if present*, uniquely identifies a weak entity instance within its identifying relationship.

# EXERCISE 1: UNIVERSITY APPLICATION—KEYS OF ENTITY TYPES

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




What kind of entity is Offering?

⟹ Weak entity dependent on Course.

Is there a discriminator for Offering?

⟹ Yes — section, semester, year.

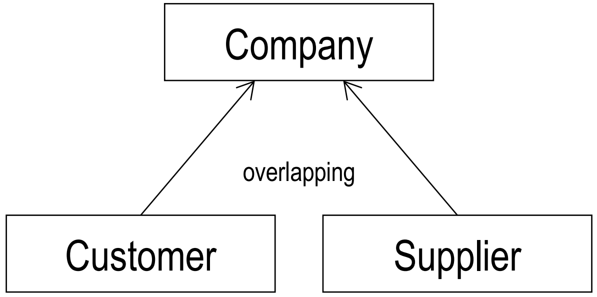
# ENTITY GENERALIZATION CONSTRAINTS: COVERAGE

## Disjointness

### (a) overlapping

A superclass instance can relate to more than one subclass.

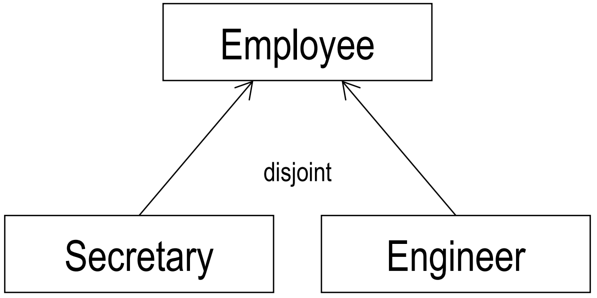
E.g., a given company can be both a customer and a supplier at the same time.




### b) disjoint

A superclass instance can relate to at most one subclass.

E.g., a given employee can be either a secretary or an engineer, but not both at the same time.



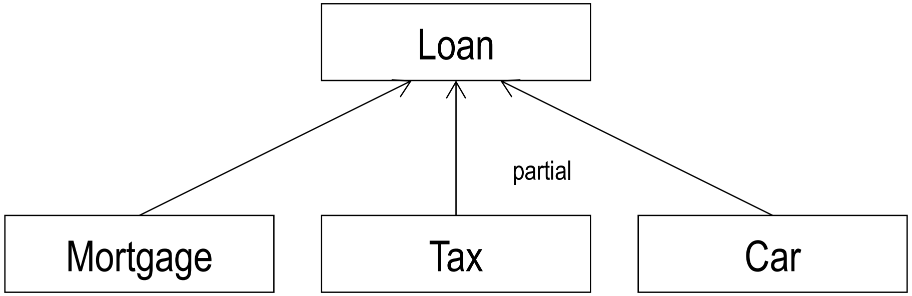
# ENTITY GENERALIZATION CONSTRAINTS: COVERAGE (cont'd)

## Completeness

### a) partial

A superclass instance does not need to relate to any of the subclasses.

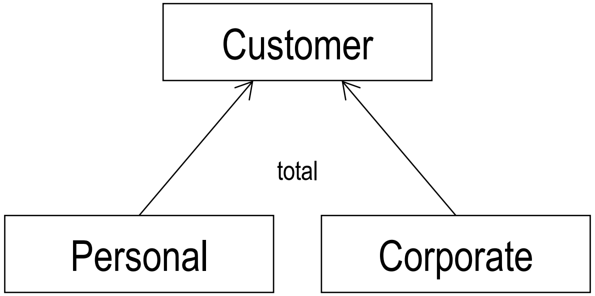
E.g., a loan does not need to be a mortgage (loan) or a tax (loan) or a car (loan)—there are other kinds of loans.




### (b) total

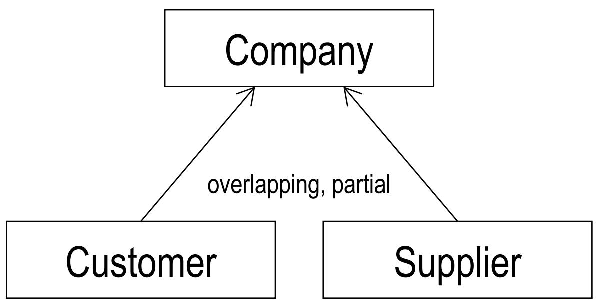
A superclass instance must relate to at least one of the subclasses.

E.g., a given customer must be either a personal or a business customer.

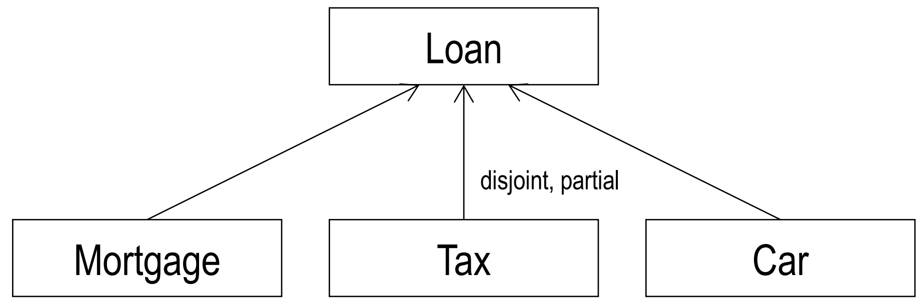


# ENTITY GENERALIZATION CONSTRAINTS: COVERAGE (cont'd)

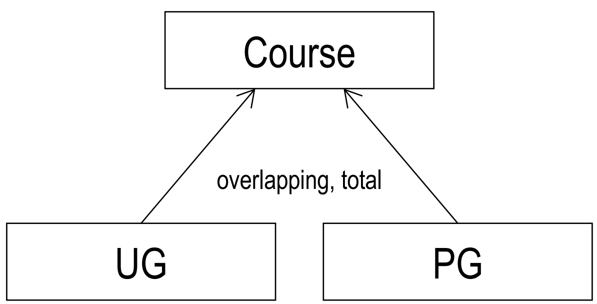
## overlapping, partial



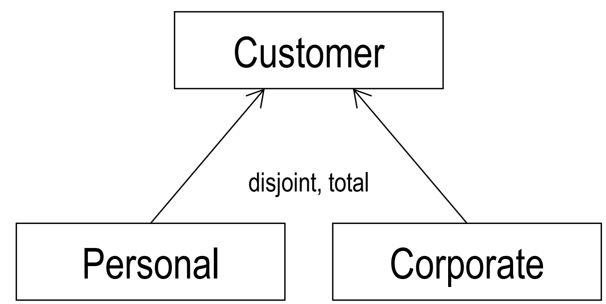

## disjoint, partial




## overlapping, total



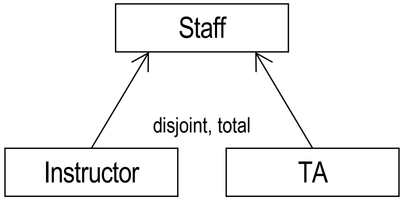

## disjoint, total




☞ **Coverage is specified as one from disjointness (when there is more than one subclass) and one from completeness.**

# EXERCISE 1: UNIVERSITY APPLICATION— ENTITY GENERALIZATION COVERAGE

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




What should be the disjointness constraint?

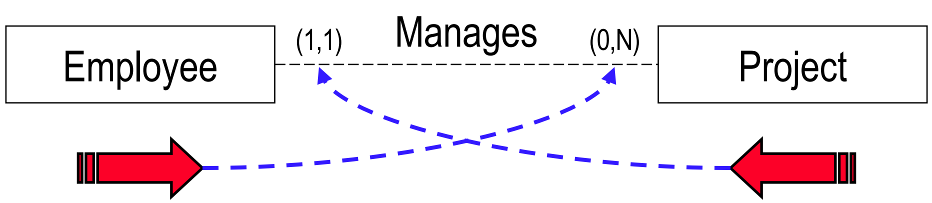
⟹ disjoint

What should be the completeness constraint?

⟹ total

# RELATIONSHIP CONSTRAINTS: CARDINALITY & PARTICIPATION

***Cardinality*** specifies the **maximum number** and ***participation*** specifies the **minimum number** of **relationship instances** in which an entity may participate.



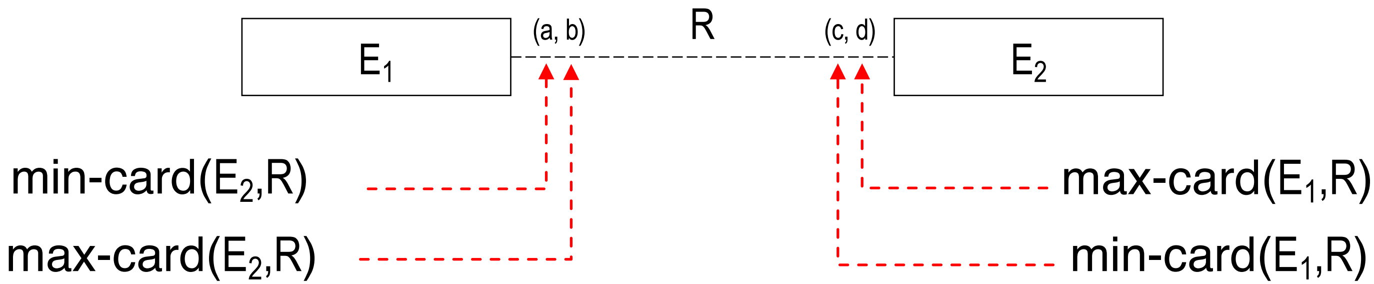

**For a given project, how many employees can manage it?**

☞ Each project is managed by one and only one employee.

**For a given employee, how many projects can he/she manage?**

☞ An employee does not have to manage any project, but may manage several (i.e., an unknown number of) projects.

# RELATIONSHIP CONSTRAINTS:
# CARDINALITY & PARTICIPATION




## minimum cardinality (min-card) ⟹ participation constraint

min-card($E_1$,R): The *minimum* number of relationship instances in which each entity of $E_1$ *must* participate in R.

min-card($E_1$,R) = 0 ⟹ partial participation

min-card($E_1$,R) > 0 ⟹ total participation

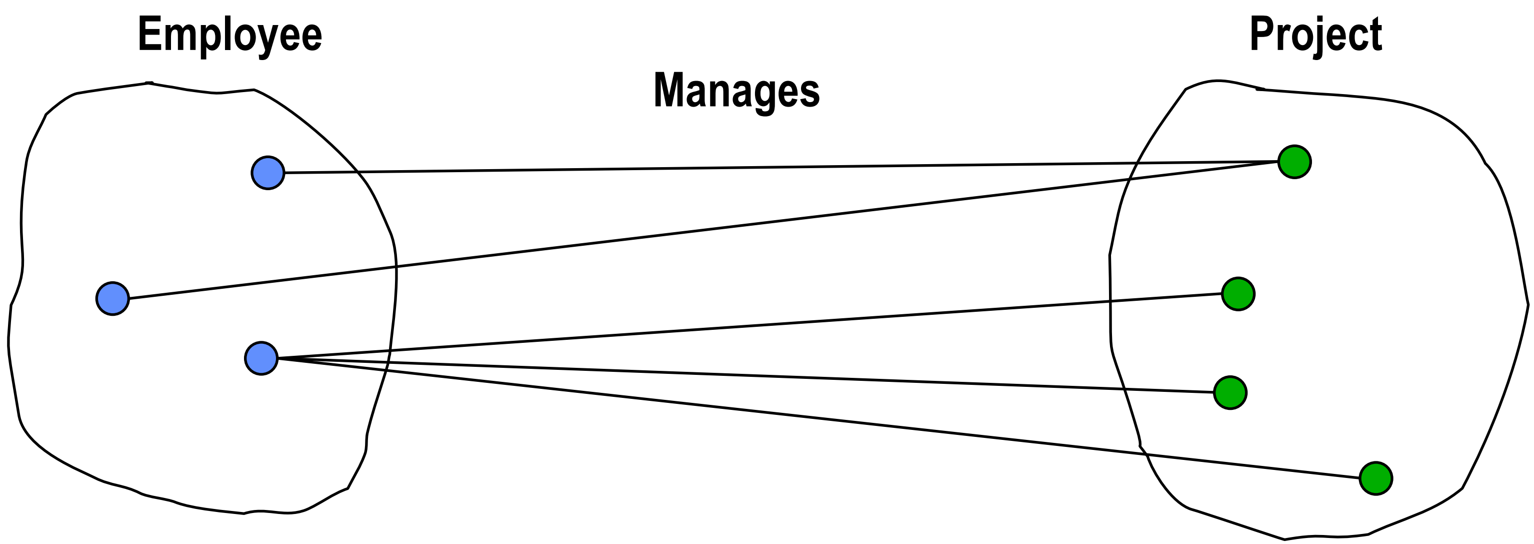
## maximum cardinality (max-card) ⟹ cardinality constraint

max-card($E_1$,R): The *maximum* number of relationship instances in which each entity of $E_1$ *may* participate in R.

# RELATIONSHIP CONSTRAINTS:
# CARDINALITY & PARTICIPATION

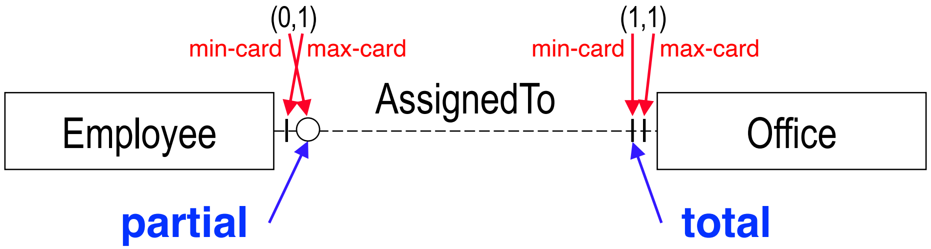

An employee does not have to manage a project but can manage several projects.

Every project must be managed by an employee and at most one employee.

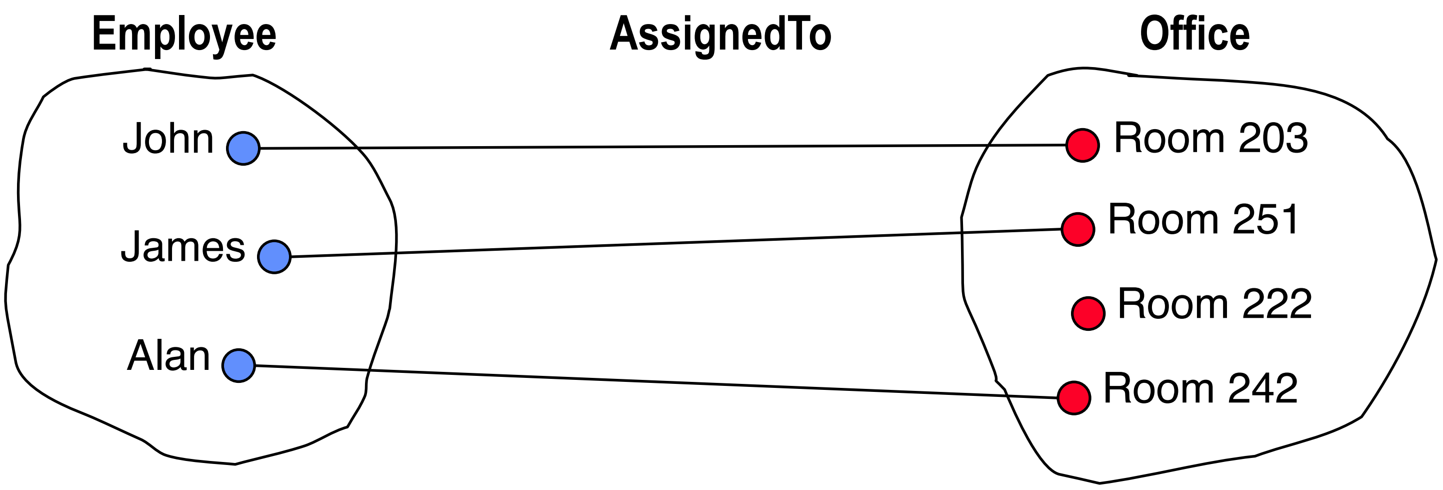


# RELATIONSHIP CONSTRAINTS:
# EXAMPLE CARDINALITY & PARTICIPATION

## one-to-one (1:1) relationship

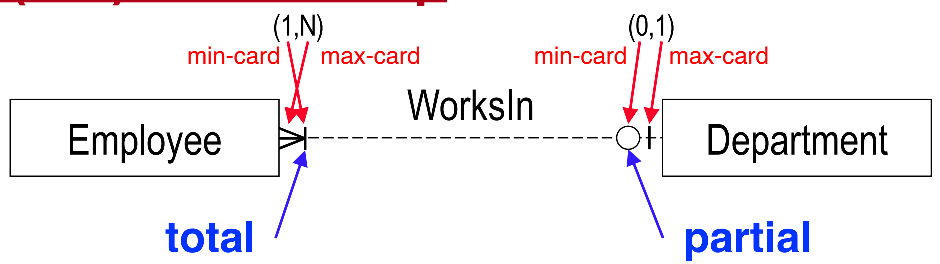


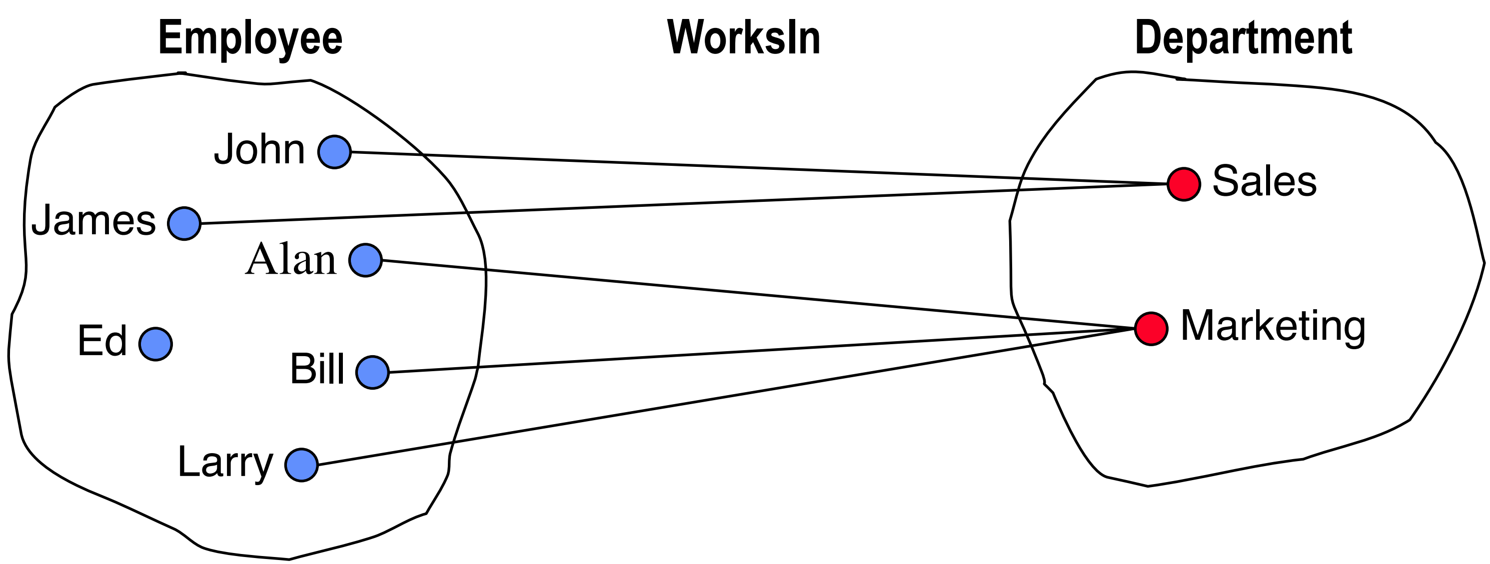

information engineering (crow foot) notation



# RELATIONSHIP CONSTRAINTS:
# EXAMPLE CARDINALITY & PARTICIPATION

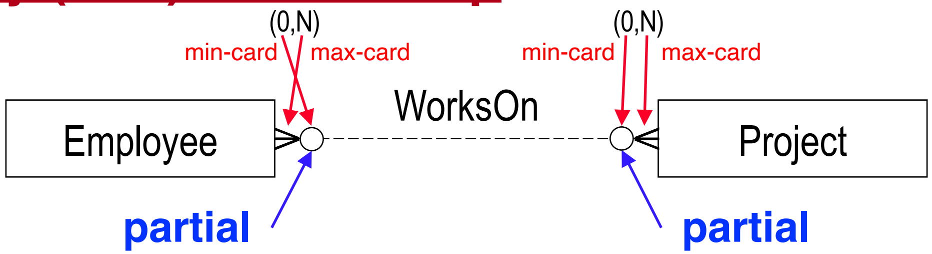
## one-to-many (1:N) relationship

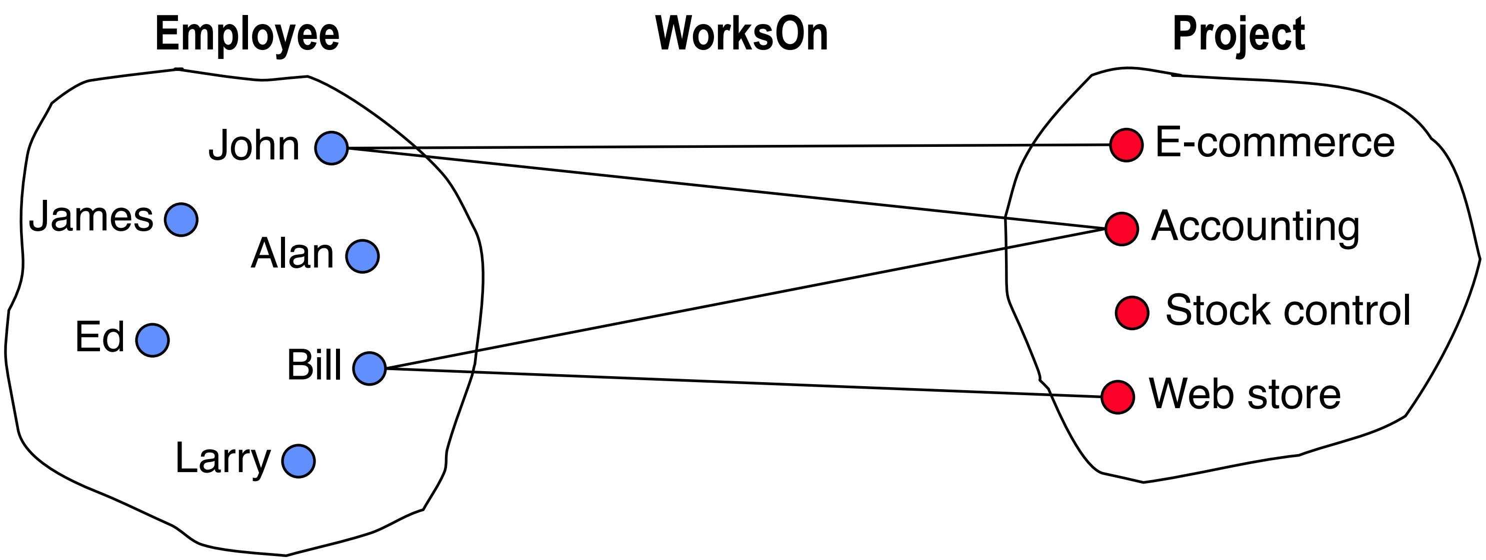


# RELATIONSHIP CONSTRAINTS:
# EXAMPLE CARDINALITY & PARTICIPATION

## many-to-many (N:M) relationship



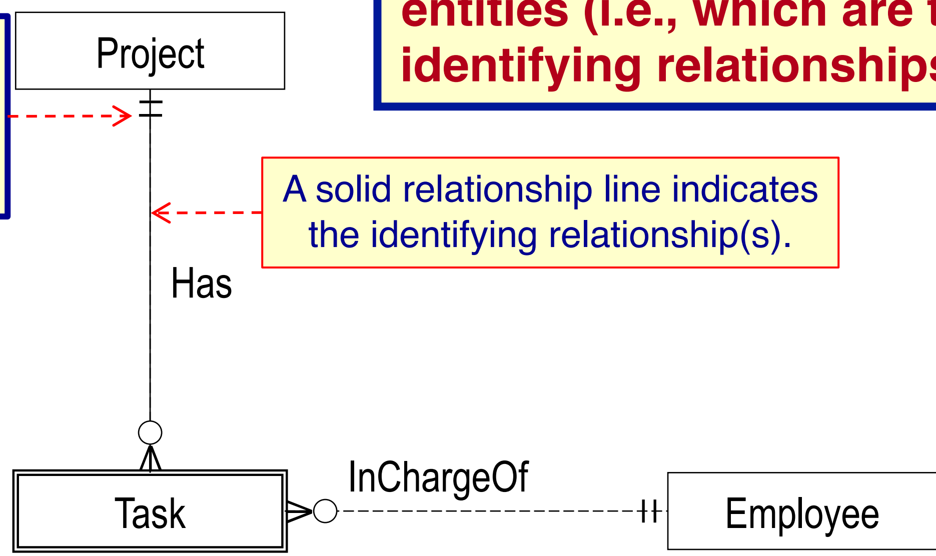
# RELATIONSHIP CONSTRAINTS:
# WEAK ENTITY PARTICIPATION

- A weak entity may be related to more than one strong entity but may depend on only some of these for its existence.

☞ **Only some of the strong entities are identifying entities.**

**How to show which entities are the identifying entities (i.e., which are the identifying relationships)?**

The participation of a weak entity in an identifying relationship is always total.




A solid relationship line indicates the identifying relationship(s).

# EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

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


What should be the cardinality constraint (max-card) for Department?

⟹ many (A department can offer many courses—domain knowledge.)

What should be the participation constraint (min-card) for Department?

⟹ unknown (Could be partial or total; need to verify with client. Leave unspecified.)

What should be the cardinality constraint (max-card) for Course?

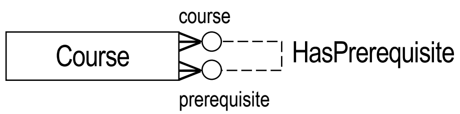
⟹ unknown (Could be 1 or N; need to verify with client. Leave unspecified.)

What should be the participation constraint (min-card) for Course?

⟹ total (Every course must be offered by some department—domain knowledge.)

# EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

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




What should be the cardinality constraints?

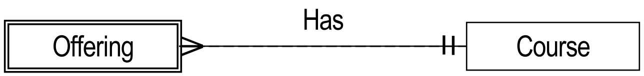
⟹ Course (prerequisite) many (A course can be a prerequisite for several courses.)
Course (course) many (A course can have several prerequisites.)

What should be the participation constraints?

⟹ Course (prerequisite) partial (A course does not have to be a prerequisite.)
Course (course) partial (A course can have no prerequisites.)

# EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

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




What should be the cardinality constraint (max-card) for Offering?

⟹ 1 (Every offering is for at most one course—domain knowledge.)

What should be the participation constraint (min-card) for Offering?

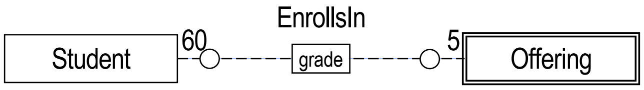
⟹ total (Every offering must be for some course—domain knowledge.)

What about for Course?

⟹ (?,many) min-card most likely 0, but need to verify with client. Leave unspecified.

# EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

- Each student must enroll in one to five course offerings.
- Each course offering can enroll zero to sixty students.




Is a student required to enroll in an offering as soon as the student's record is created?

**No!**
(domain knowledge)

Is Offering dependent on Student?

⟹ No.

What should be the cardinality constraint (max-card) for Student?

⟹ 5 (A student can enroll in at most 5 course offerings.)

What should be the participation constraint (min-card) for Student?

⟹ total (A student must enroll in at least 1 course offering.)

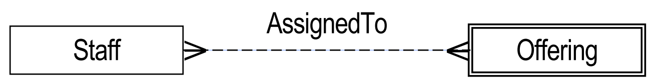
What about for Offering?

⟹ (0, 60)

☞ **Does the participation constraint for Student make sense?**

# EXERCISE 1: UNIVERSITY APPLICATION—RELATIONSHIP CARDINALITY & PARTICIPATION

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




Is an offering required to have a staff assigned to it?

**Need to verify with client!**

Is Offering dependent on Staff?

⟹ No.

What should be the cardinality constraint (max-card) for Offering?

⟹ many (An offering can have several staff assigned to it.)

What should be the participation constraint (min-card) for Offering?

⟹ total (An offering has at least one staff assigned to it.)

What about for Staff?

⟹ (?,many) min-card most likely 0, but need to verify with client. Leave unspecified.

☞ **Does the participation constraint for Offering make sense?**

# EXERCISE 1: UNIVERSITY APPLICATION— RELATIONSHIP CARDINALITY & PARTICIPATION

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


What should be the cardinality constraint (max-card) for Staff?

⟹ 1 (For each staff … we store the … department ….)

What should be the participation constraint (min-card) for Staff?

⟹ total (Every staff must be appointed in some department—domain knowledge.)

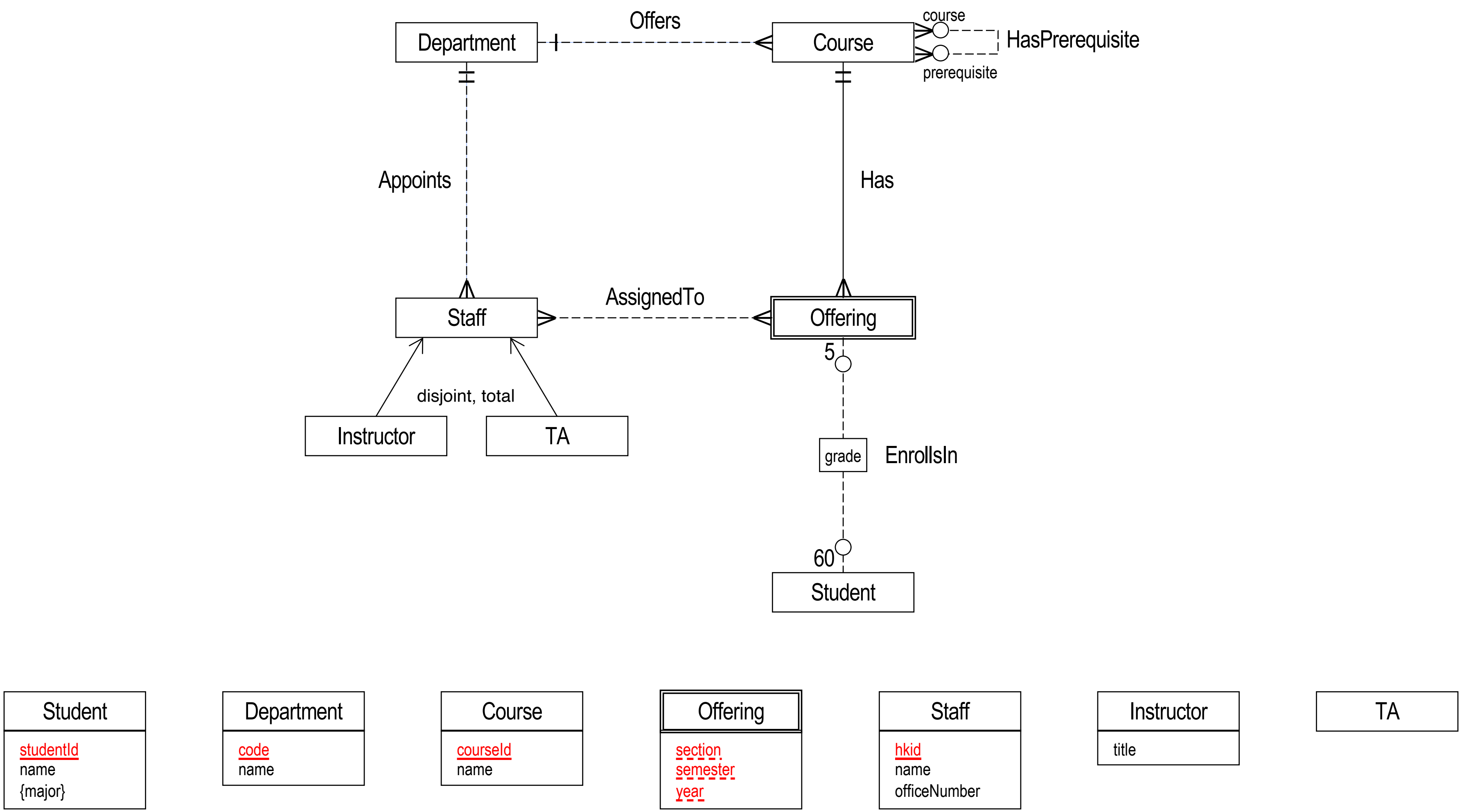
What should be the cardinality constraint (max-card) for Department?

⟹ many (A department can appoint several staff—domain knowledge.)

What should be the participation constraint (min-card) for Department?

⟹ unknown (Could be partial or total; need to verify with client. Leave unspecified.)

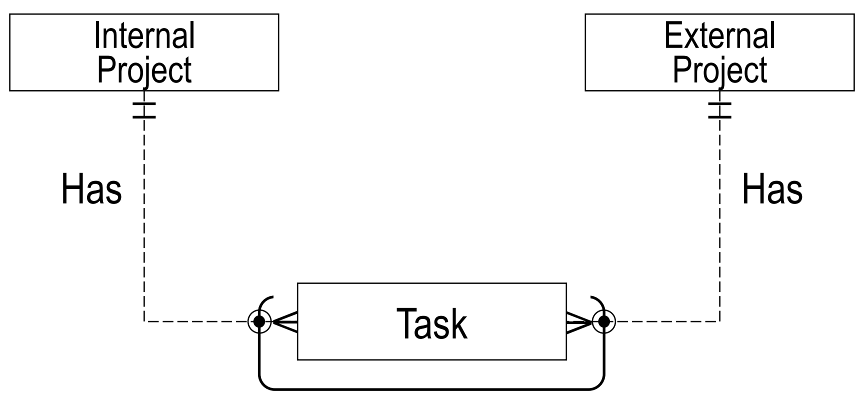
# EXERCISE 1: UNIVERSITY APPLICATION— E-R DIAGRAM



# RELATIONSHIP CONSTRAINTS: EXCLUSION

**An *exclusion (XOR) constraint* specifies that at most one entity instance, among several entity types, can participate in a relationship with a single "root" entity.**

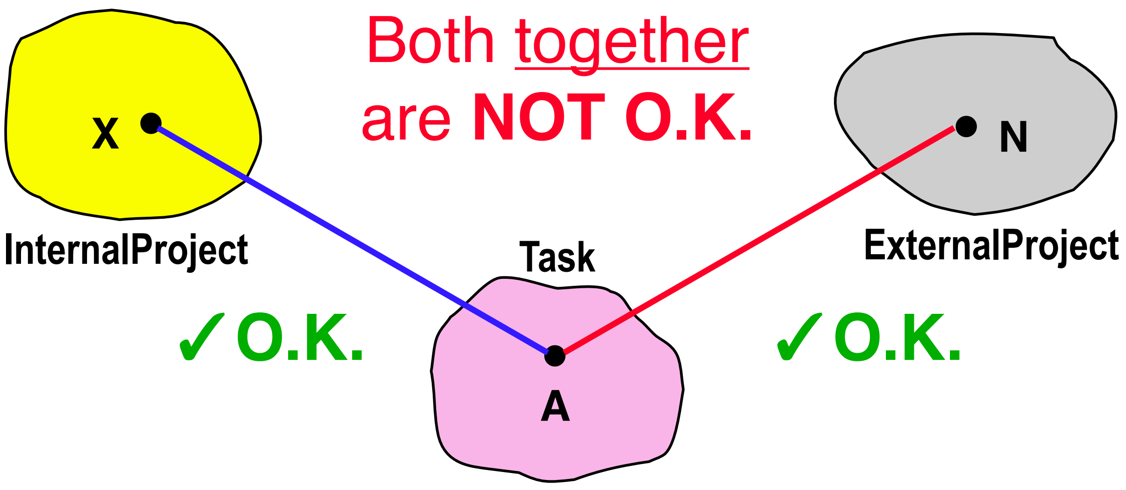
**Example:** A task can be related to *either* an internal project *or* an external project, *but not both*.



# RELATIONSHIP CONSTRAINTS: EXCLUSION (cont'd)

**An *exclusion (XOR) constraint* specifies that at most one entity instance, among several entity types, can participate in a relationship with a single "root" entity.**

**Example:** A task can be related to *either* an internal project *or* an external project, *but not both*.



# E-R MODEL & DB DESIGN: OUTLINE

✓ Database Design Process

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

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

➡ **Analyzing Application Requirements / Making Design Choices**

Reduction of E-R Schemas to Relational Schemas

# ANALYZING APPLICATION REQUIREMENTS

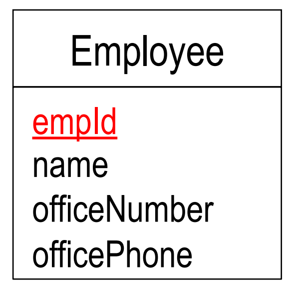
1. **Identify entities**
   - What are the major concepts about which data needs to be permanently stored?
   - Focus on the "big picture", not the details.
     - E.g., student, course not name, address, email, description, credits, etc.
2. **Identify relationships between entities**
   - How are the major concepts related? How do they interact?
   - What interactions need to be permanently stored.
     - E.g., students enroll in courses not students browse courses
3. **Identify properties of entities and relationships**
   - For each entity and relationship, what information needs to be permanently stored.

# DESIGN CHOICE: ENTITY VERSUS ATTRIBUTE

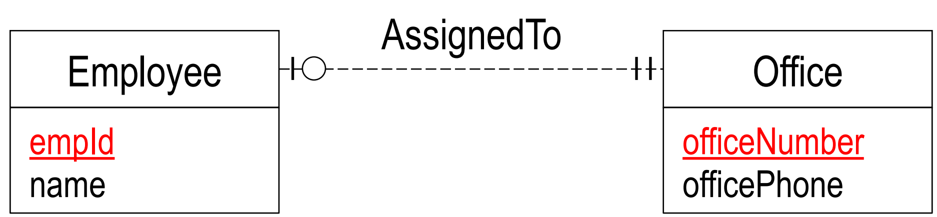
entity: When several properties can be associated with the concept.

attribute: When the concept has a simple atomic structure or no property of interest.

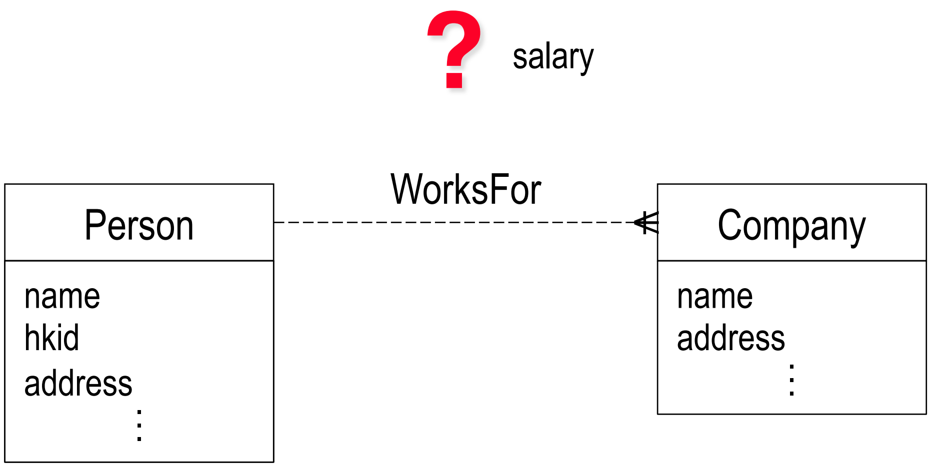
## Office as attribute




## Office as entity



# DESIGN CHOICE: PLACING AN ATTRIBUTE




**Where to place salary?**

**Relationship attributes are usually needed only for many to many relationships!**
**(But can also be used in one to one and one to many relationships.)**

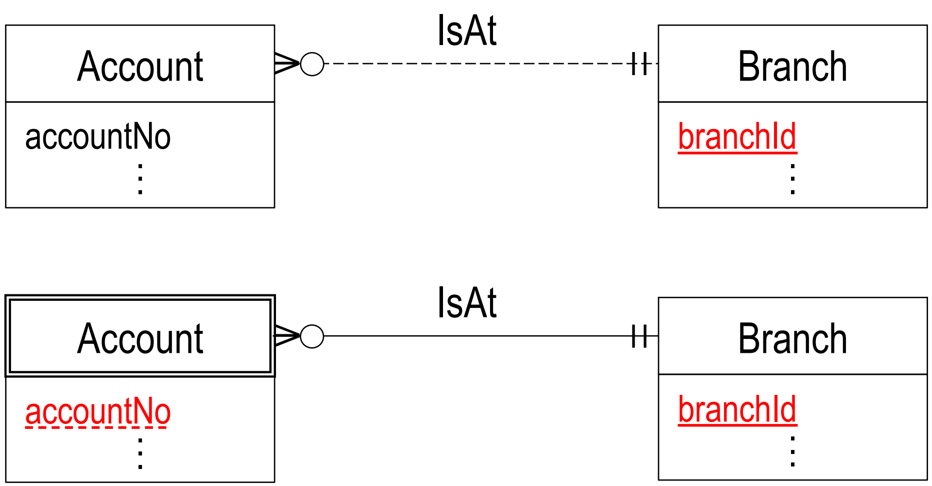
# DESIGN CHOICE: STRONG VERSUS WEAK ENTITY

strong entity: When the concept can be uniquely identified in the application domain (i.e., it has a key).

weak entity: When the concept has no unique identifier.

**Suppose an account must be associated with exactly one branch and two different branches can have accounts with the same number.**

**Should Account be a strong or weak entity?**

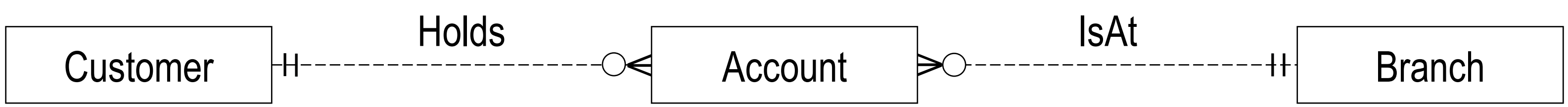


# DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP

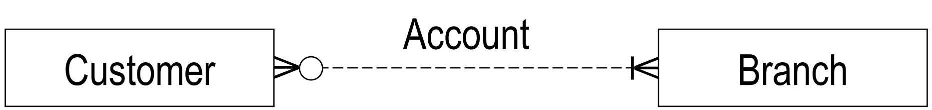
entity: When the concept represents something distinct in the application domain with several properties.

relationship: When the concept is not a distinct application domain concept and/or has no property of interest.

## Account as entity



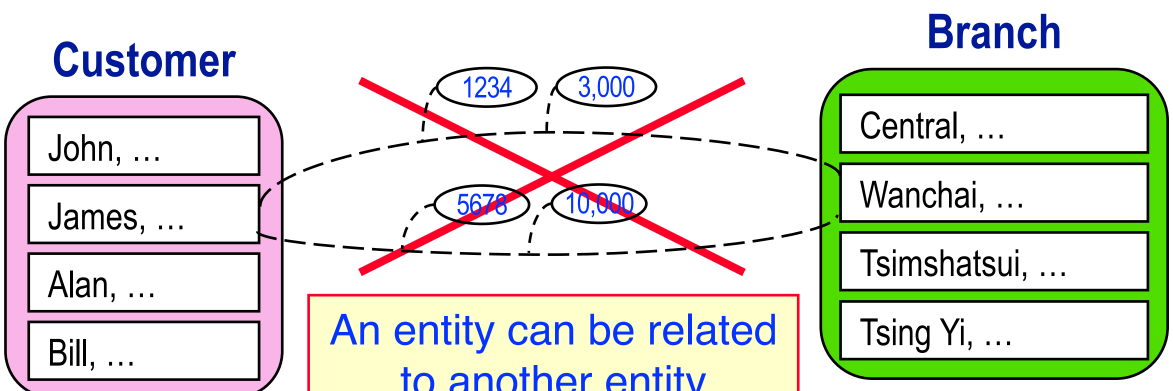

## Account as relationship




**What if you want to have several accounts for a customer at the same branch?**

# DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP (cont'd)

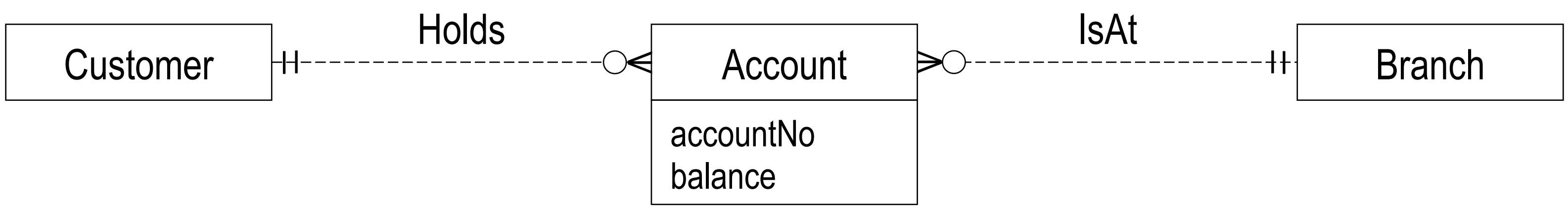
**We want to represent the fact that James has two accounts at the same branch.**

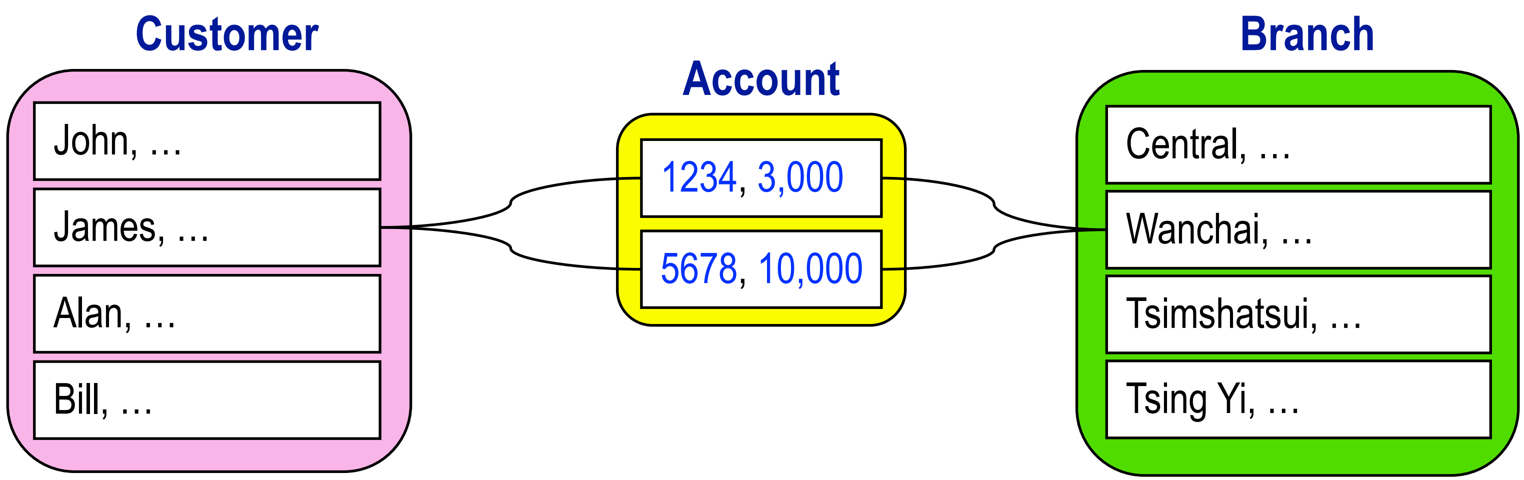




An entity can be related to another entity *at most once* by a given relationship.

# DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP (cont'd)

**We need to use an entity for Account!**




**There can be only one relationship instance of a given relationship type between the same two entity instances.**

# E-R MODEL & DATABASE DESIGN

# EXERCISE 2

Upload your completed exercise worksheet to Canvas by **11 p.m.**