COMP 3311
DATABASE MANAGEMENT SYSTEMS

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

E-R MODEL: CONSTRAINTS

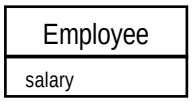
A constraint is a logical restriction or property of data that for any set of data values:

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

Examples:

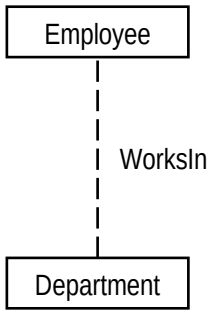
on attributes

salary is between $0 and $100,000

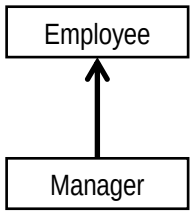


on relationships

every employee works in *at most* one department



not every employee is a manager

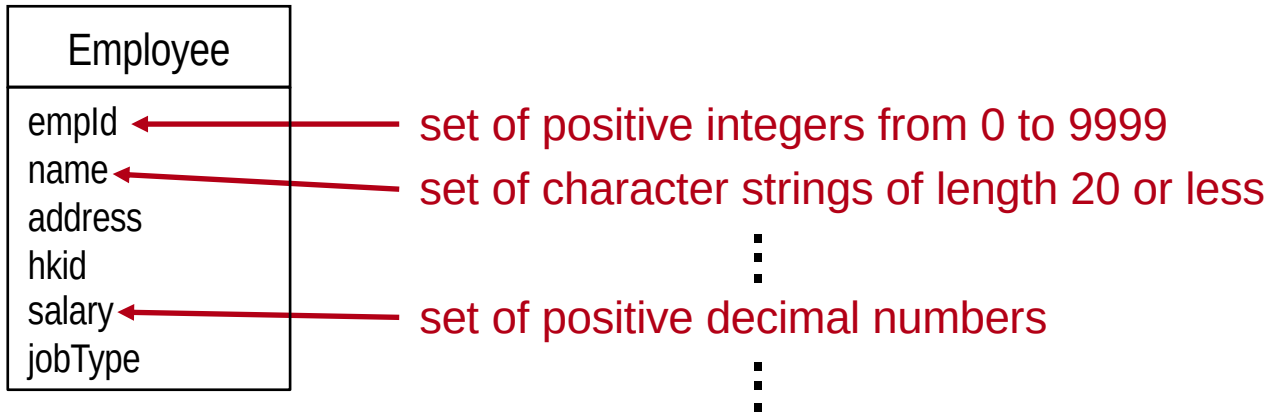


☞ **Constraints add additional semantics (meaning) to data**

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

ATTRIBUTE CONSTRAINTS: DOMAIN

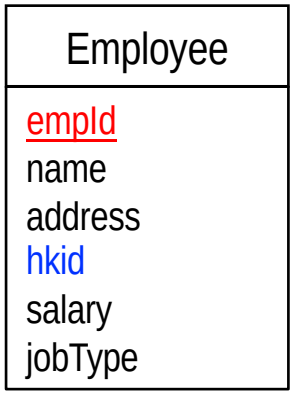
A *domain constraint* restricts an attribute to only have certain values.



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

ATTRIBUTE CONSTRAINTS: KEY

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



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

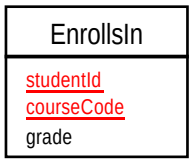
- One candidate key is selected to be the primary key.

  ☞ **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

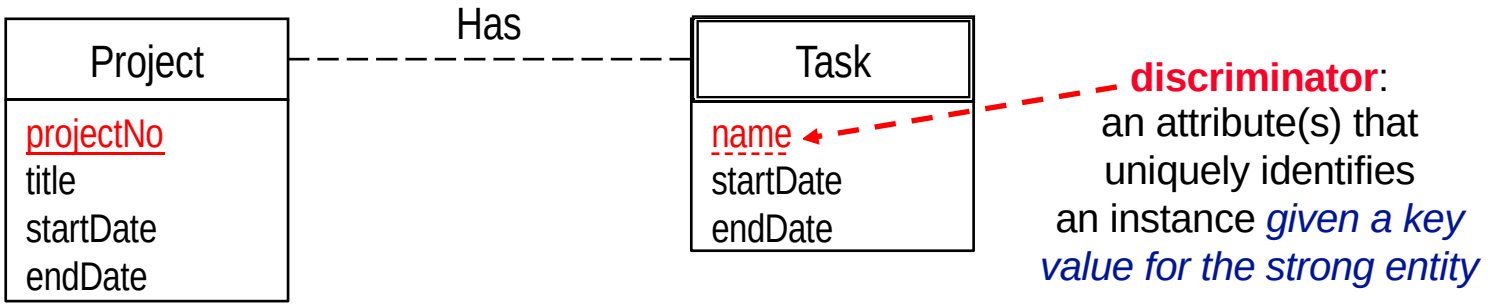
- A candidate/primary key can be composed of a set of attributes ⟹ composite key.



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.
- A discriminator, *if present*, uniquely identifies a weak entity instance within its identifying relationship.

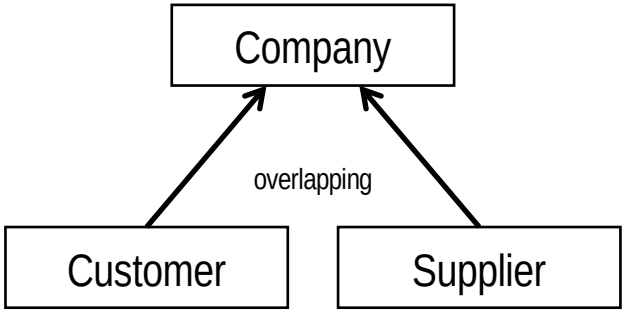
GENERALIZATION CONSTRAINTS: COVERAGE

Disjointness

(a) overlapping

A superclass instance can relate to more than one subclass.

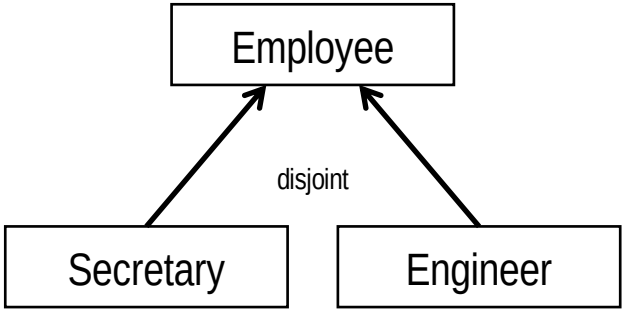
E.g., a given company can be both a customer and a supplier at the same time.



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.



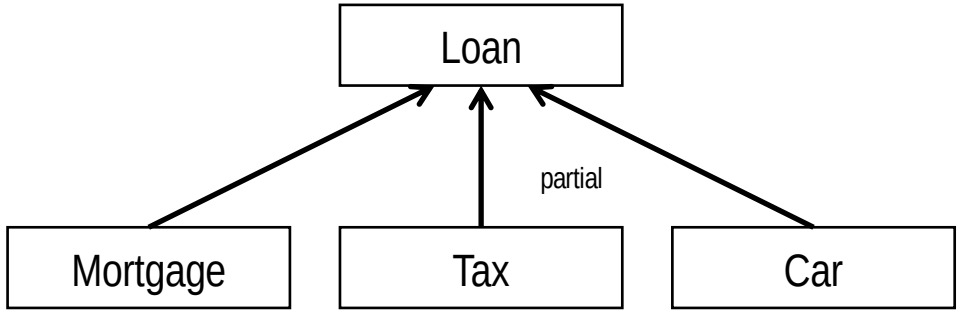
GENERALIZATION CONSTRAINTS: COVERAGE (CONT'D)

Completeness

a) partial

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

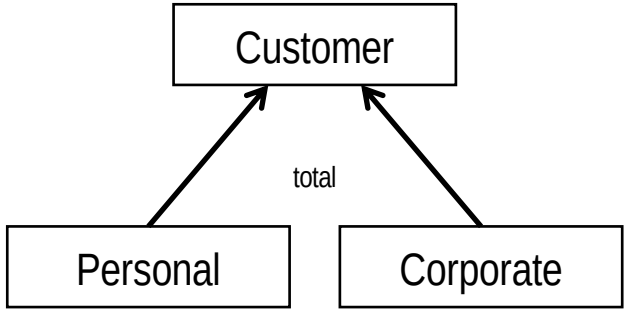
E.g., a loan does not need to be a mortgage (loan) or a tax (loan) or a car (loan)—there are other kinds of loans.



(b) total

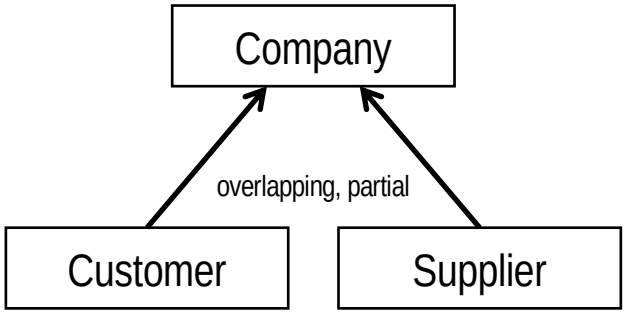
A superclass instance must relate to at least one of the subclasses.

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

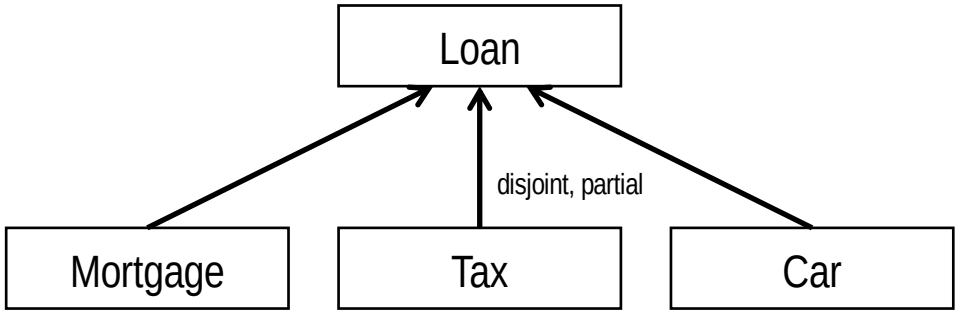


GENERALIZATION CONSTRAINTS: COVERAGE (CONT'D)

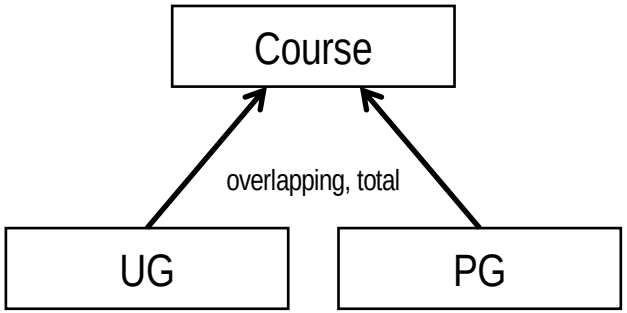
overlapping, partial



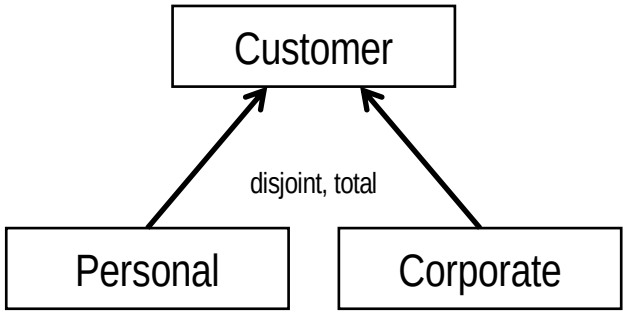
disjoint, partial



overlapping, total

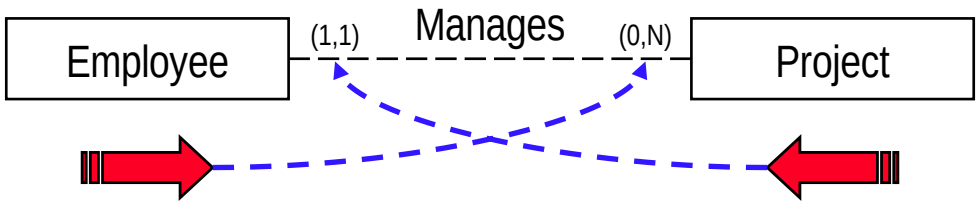


disjoint, total



RELATIONSHIP CONSTRAINTS: CARDINALITY & PARTICIPATION

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



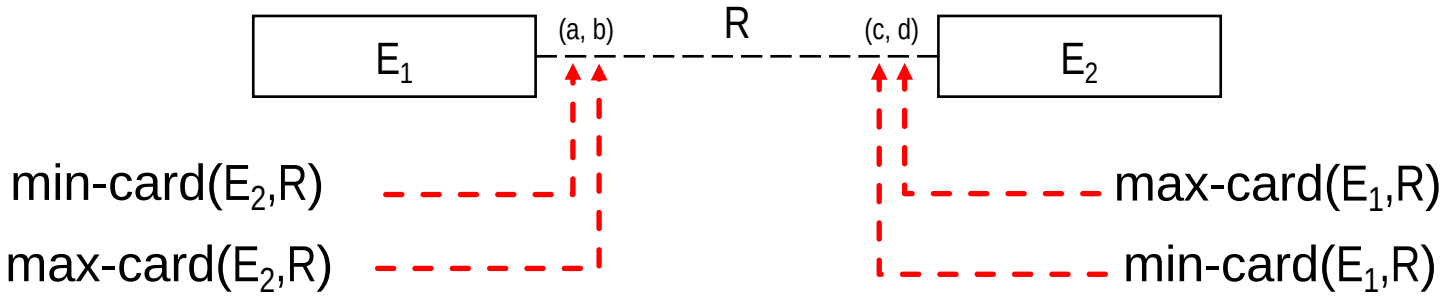
For a given project, how many employees can manage it?

- ☞ 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:
SPECIAL MAXIMUM CARDINALITIES

max-card(E_1,R) = 1 and max-card(E_2,R) = 1

- ☞ **one-to-one relationship (1:1)**

max-card(E_1,R) = 1 and max-card(E_2,R) = N

- ☞ **one-to-many relationship (1:N)**

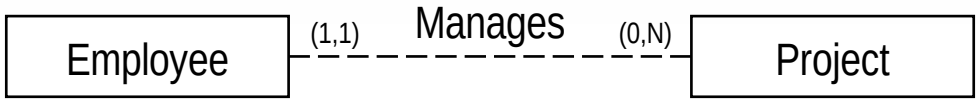
max-card(E_1,R) = N and max-card(E_2,R) = 1

- ☞ **many-to-one relationship (N:1)**

max-card(E_1,R) = N and max-card(E_2,R) = N

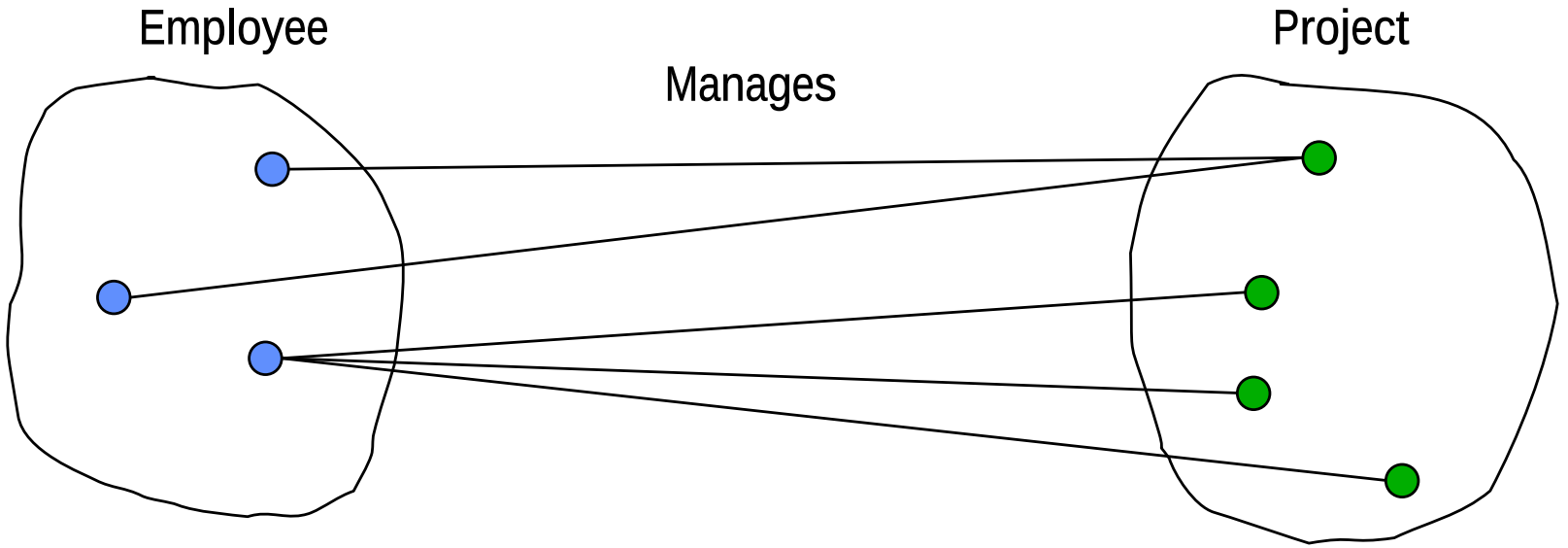
- ☞ **many to many relationship (N:M)**

RELATIONSHIP CONSTRAINTS:
CARDINALITY & PARTICIPATION



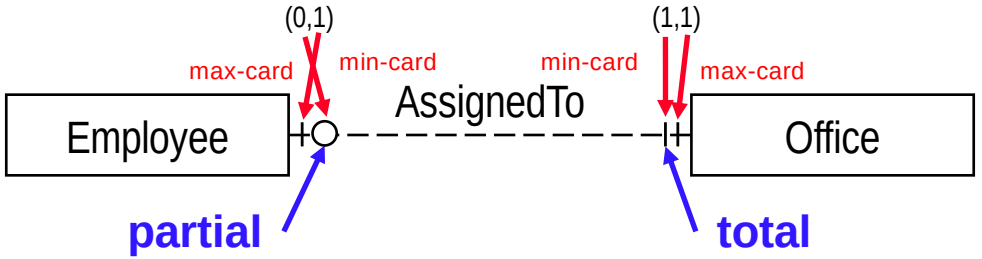
An employee does not have to manage a project, but can manage several projects.

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

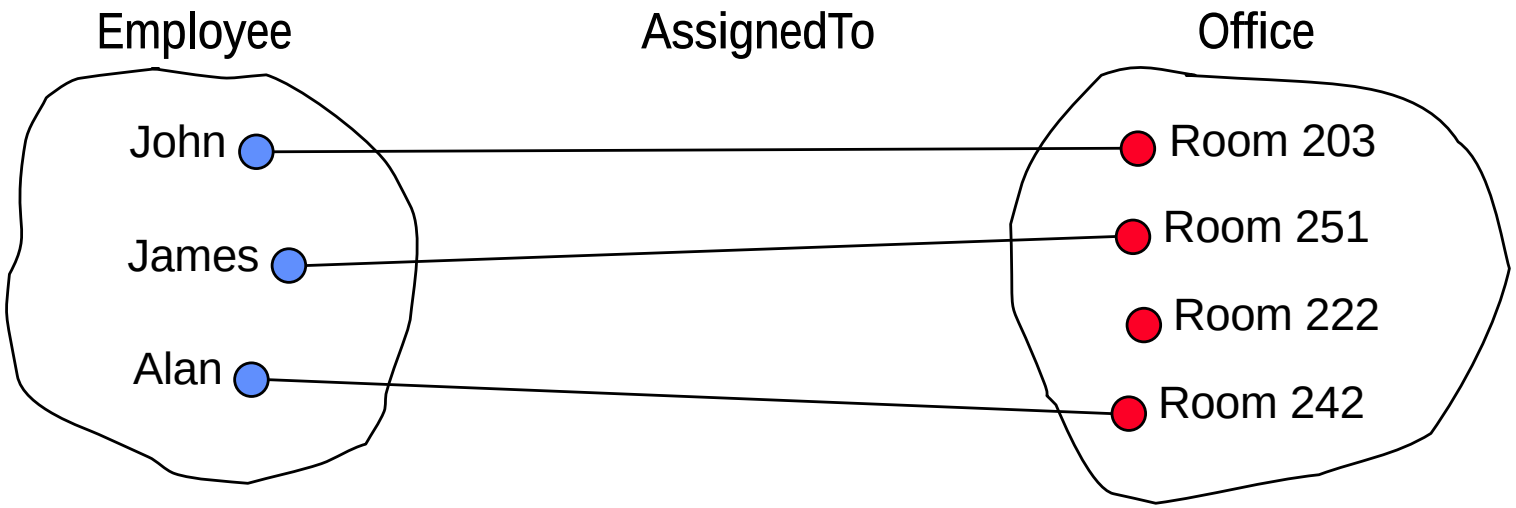


RELATIONSHIP CONSTRAINTS:
EXAMPLE CARDINALITY & PARTICIPATION

one-to-one (1:1) relationship

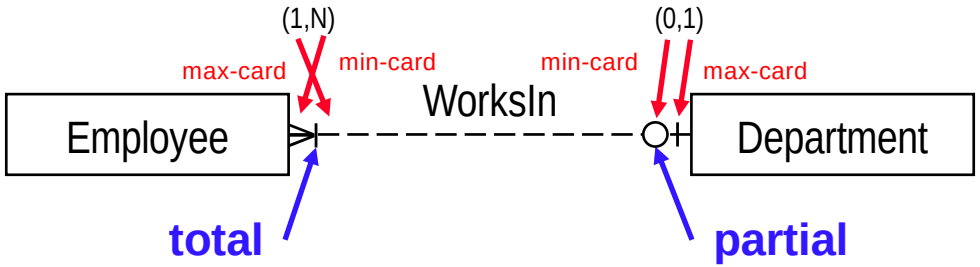


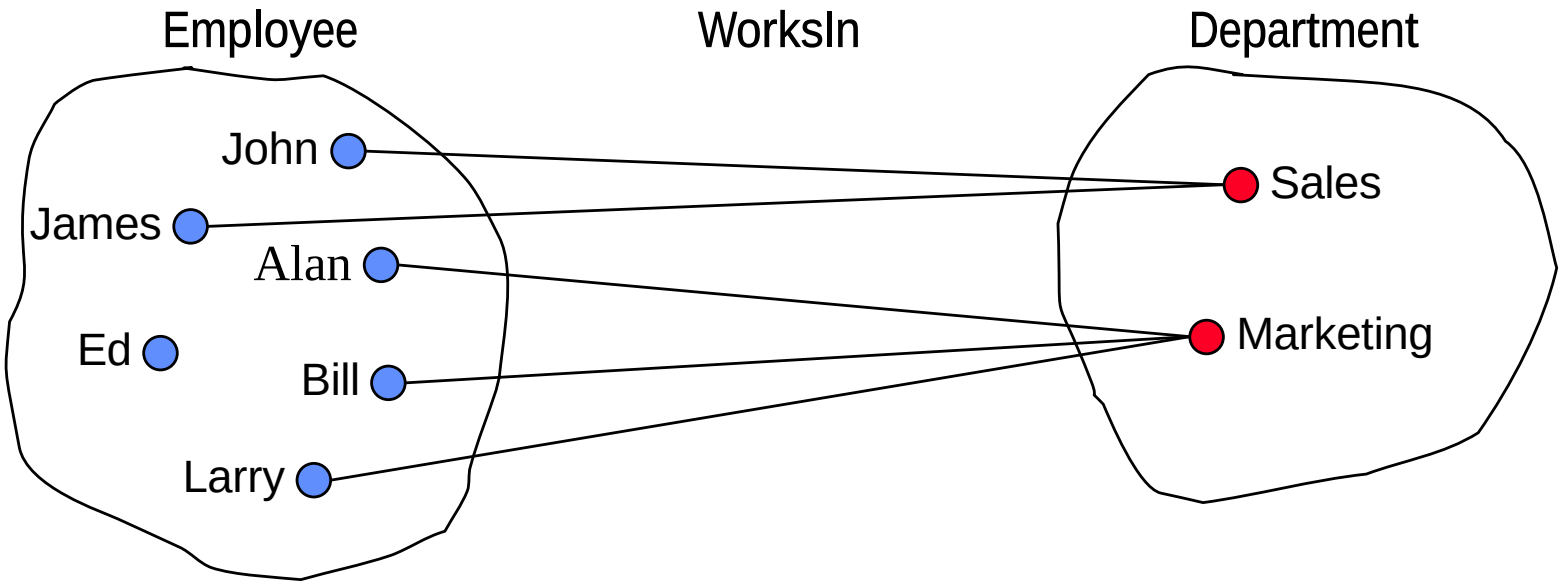
information engineering (crows foot) notation



RELATIONSHIP CONSTRAINTS:
EXAMPLE CARDINALITY & PARTICIPATION

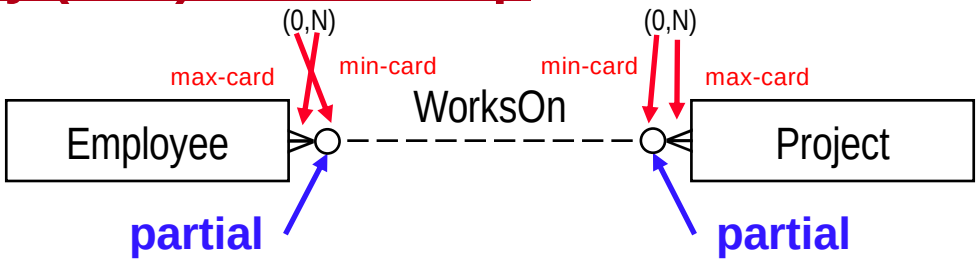
one-to-many (1:N) relationship

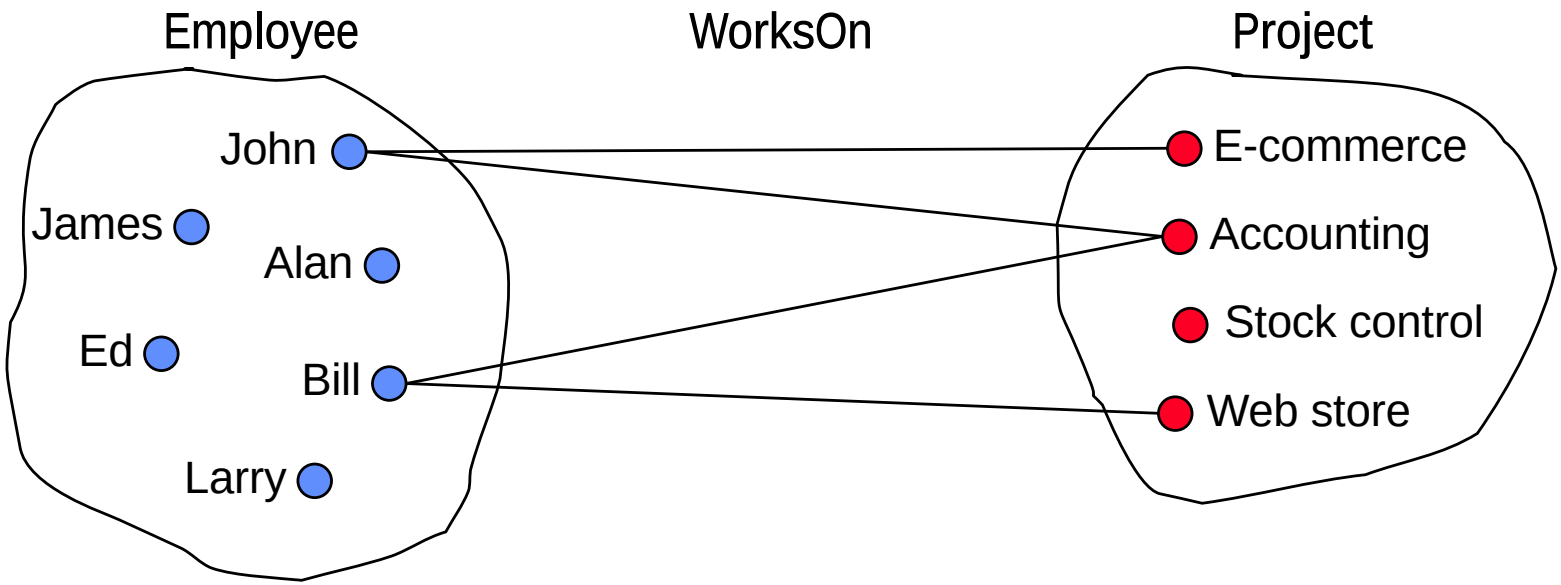




RELATIONSHIP CONSTRAINTS:
EXAMPLE CARDINALITY & PARTICIPATION

many-to-many (N:M) relationship

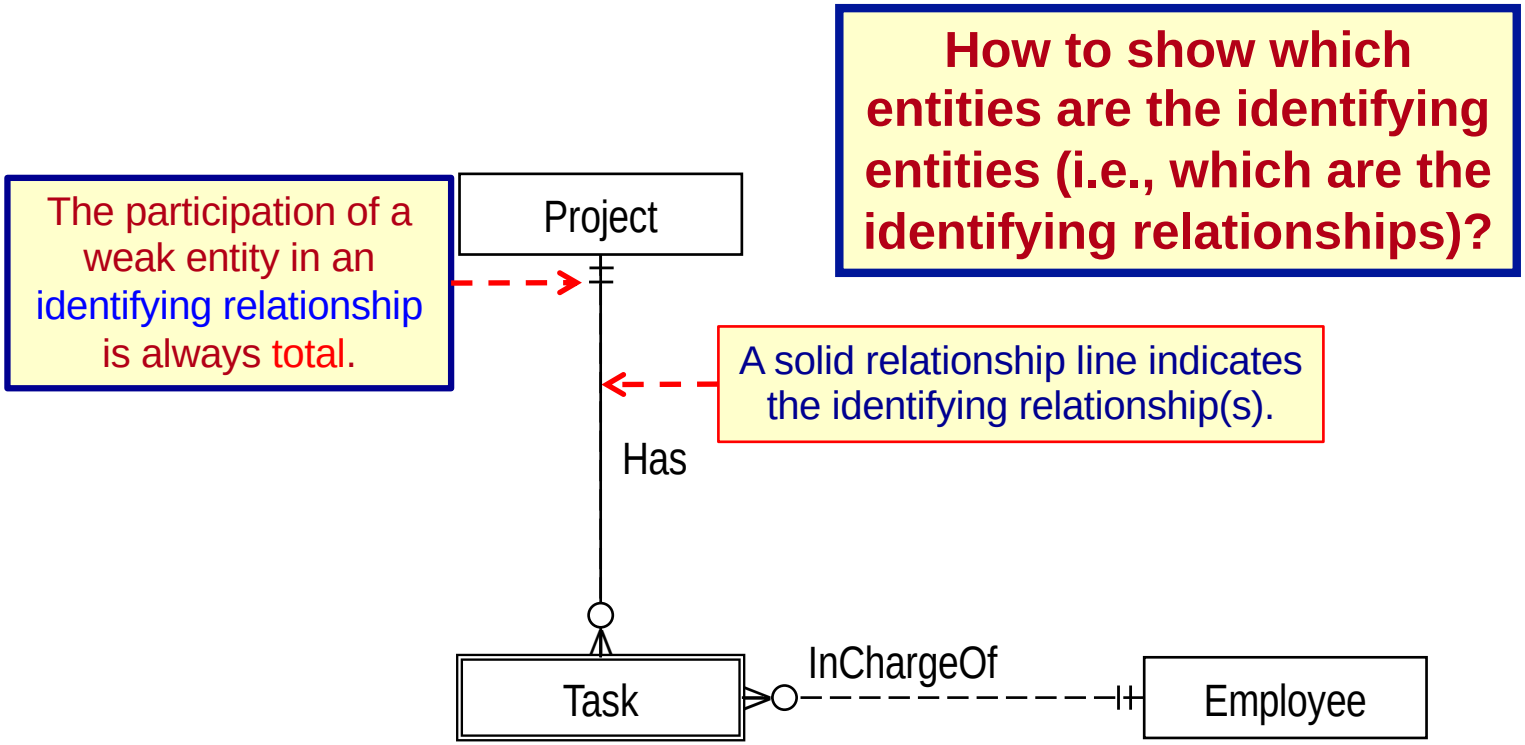




RELATIONSHIP CONSTRAINTS: WEAK ENTITY PARTICIPATION

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

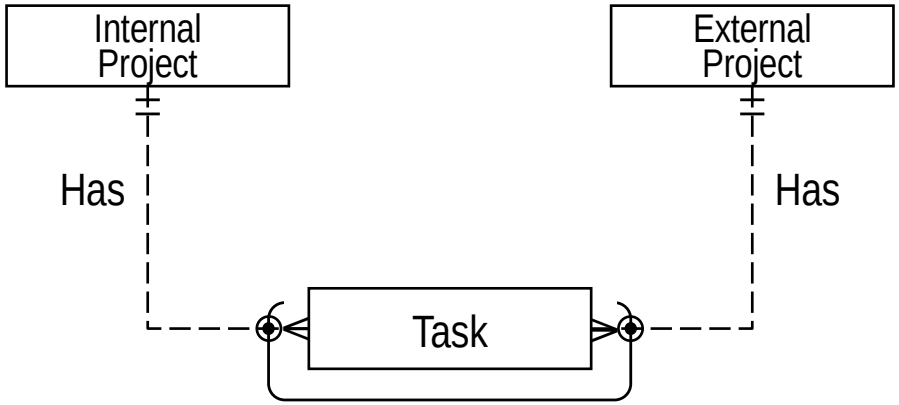
☞ **Only some of the strong entities are identifying entities.**



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.

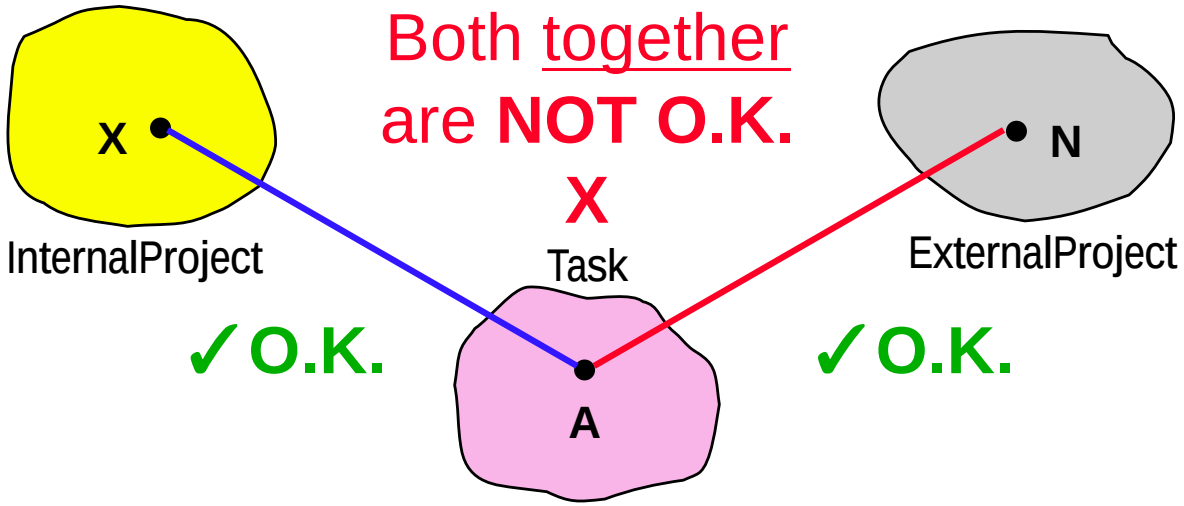
Example: A task can be related to *either* an internal project *or* an external project, *but not both*.



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*.



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

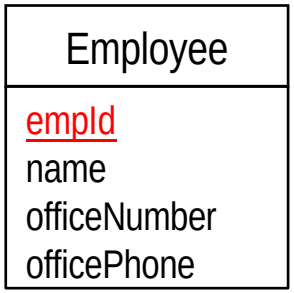
- For each entity and relationship, what information needs to be permanently stored.

DESIGN CHOICE: ENTITY VERSUS ATTRIBUTE

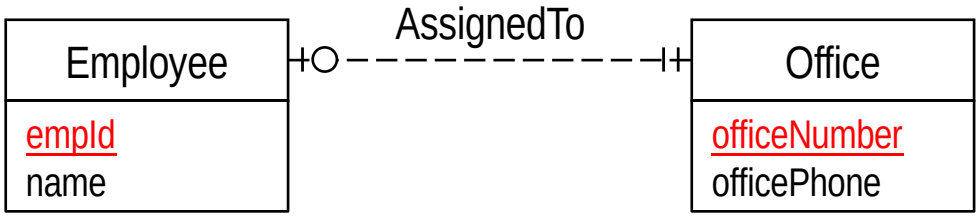
entity: When several properties can be associated with the concept.

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

Office as attribute



Office as entity

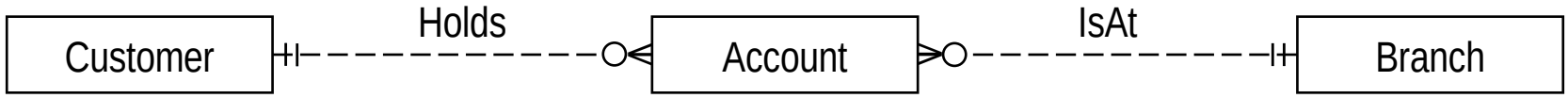


DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP

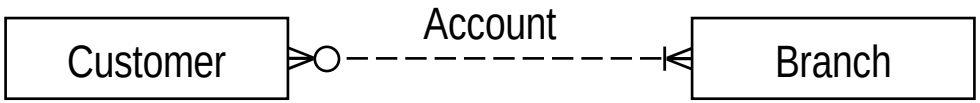
entity: When the concept represents something distinct in the application domain with several properties.

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

Account as entity



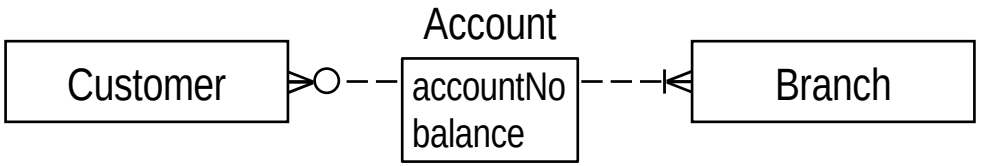
Account as relationship

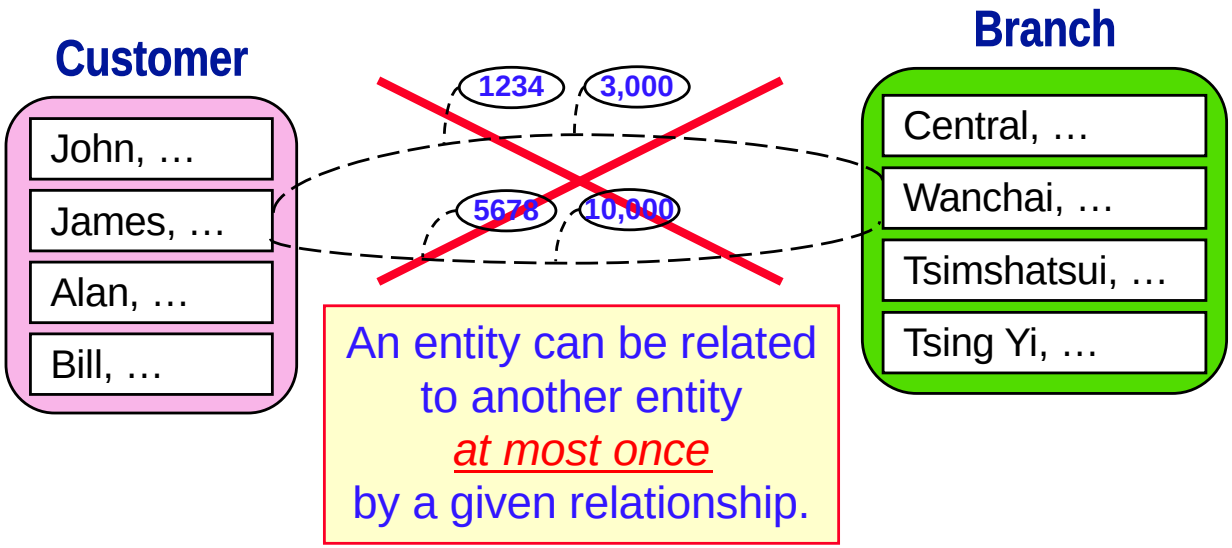


What if you want to have several accounts for a customer?

DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP (CONT'D)

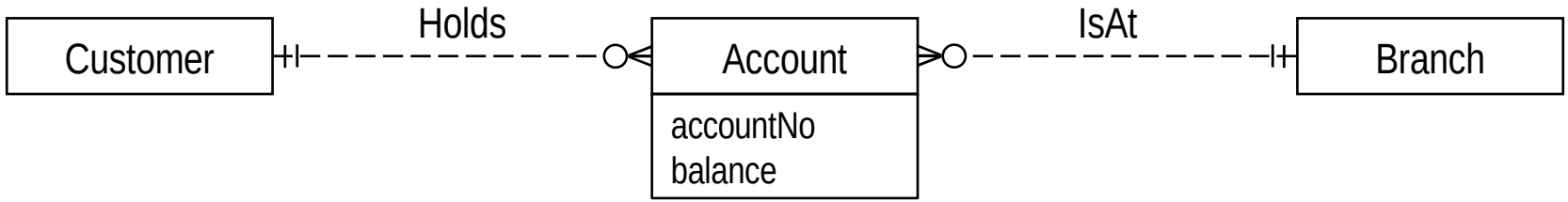
We want to represent the fact that James has two accounts at the same branch.

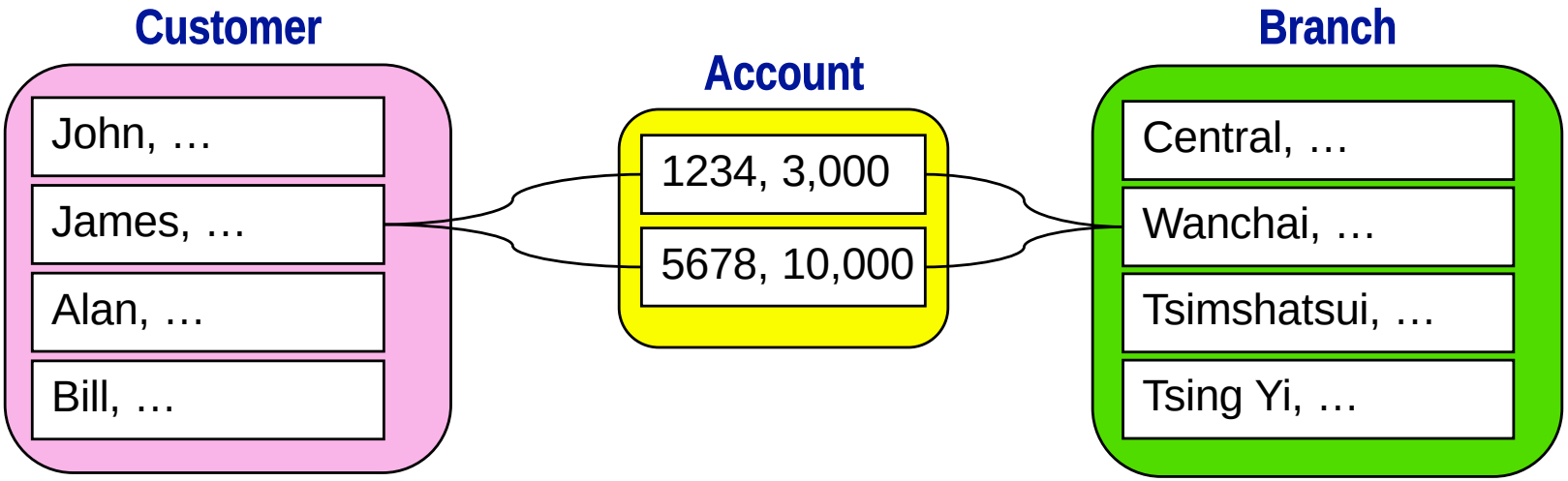




DESIGN CHOICE: ENTITY VERSUS RELATIONSHIP (CONT'D)

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.

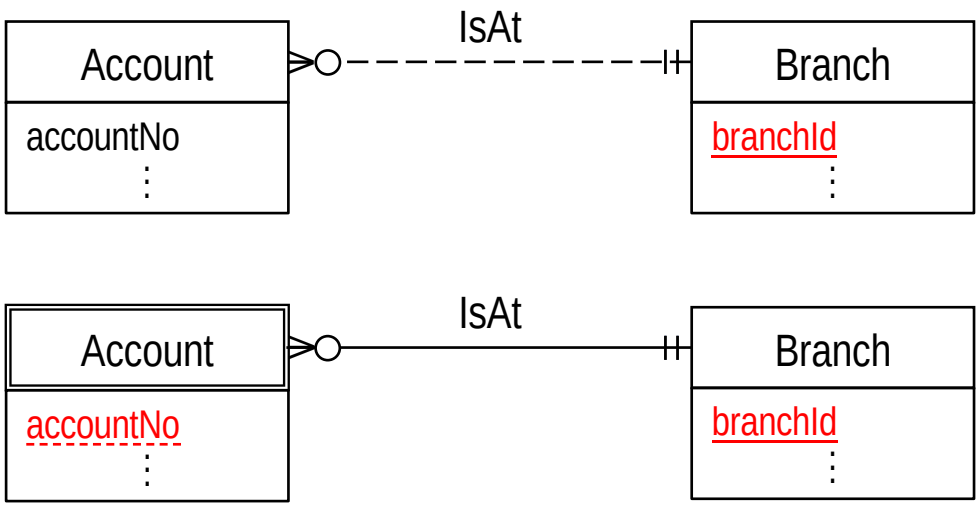
DESIGN CHOICE: STRONG VERSUS WEAK ENTITY

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

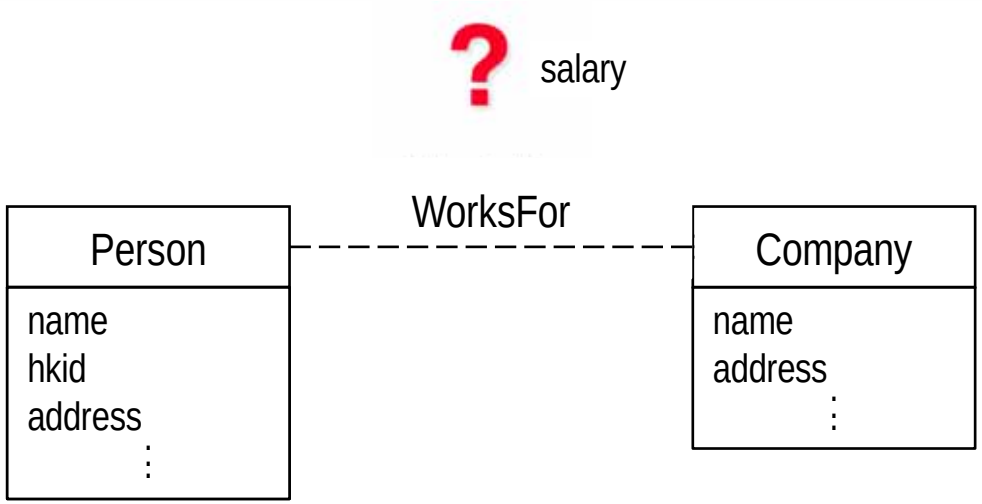
weak entity: When the concept has no unique identifier.

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

Should Account be a strong or weak entity?



DESIGN CHOICE: PLACING AN ATTRIBUTE



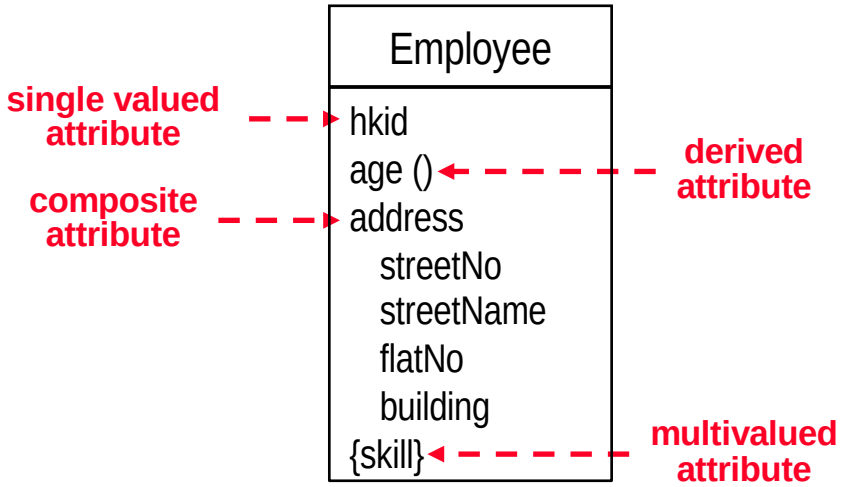
Where to place salary?

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.)

E-R MODEL & DB DESIGN: SUMMARY

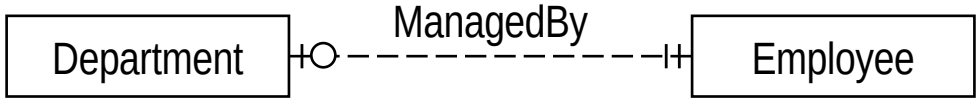
Entity-Relationship Model

entity and attribute

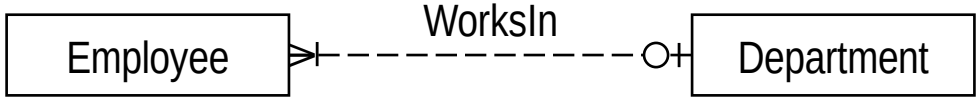


relationship ○ - partial (min-card=0) | - total (min-card=1)

(a) one-to-one (1:1)

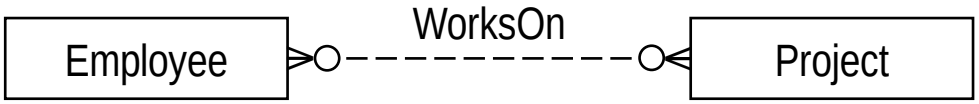


(b) one-to-many (1:N)

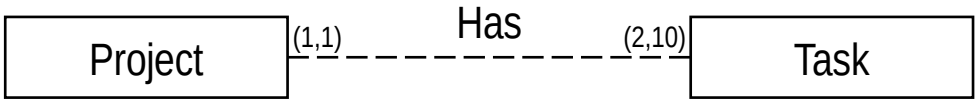


E-R MODEL & DB DESIGN: SUMMARY (CONT'd)

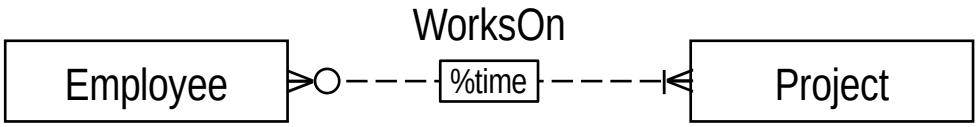
(c) many-to-many (N:M)



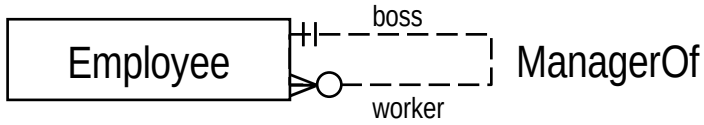
(d) explicit cardinality



(e) relationship attribute

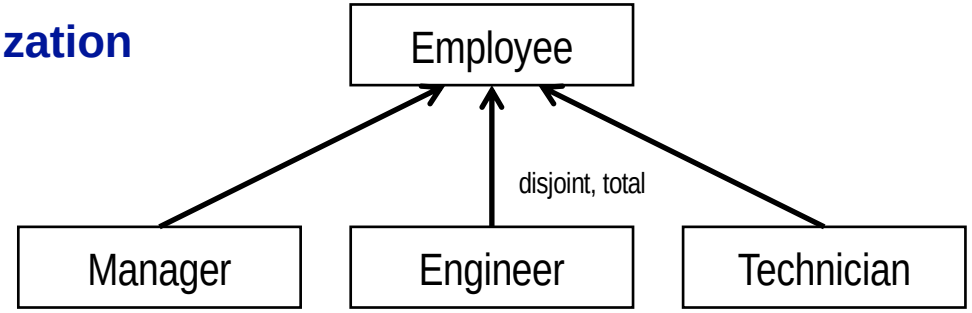


(f) recursive (with role names)



E-R MODEL & DB DESIGN: SUMMARY (CONT'd)

(h) generalization/specialization



(i) exclusion constraint