COMP 3311 DATABASE MANAGEMENT SYSTEMS

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

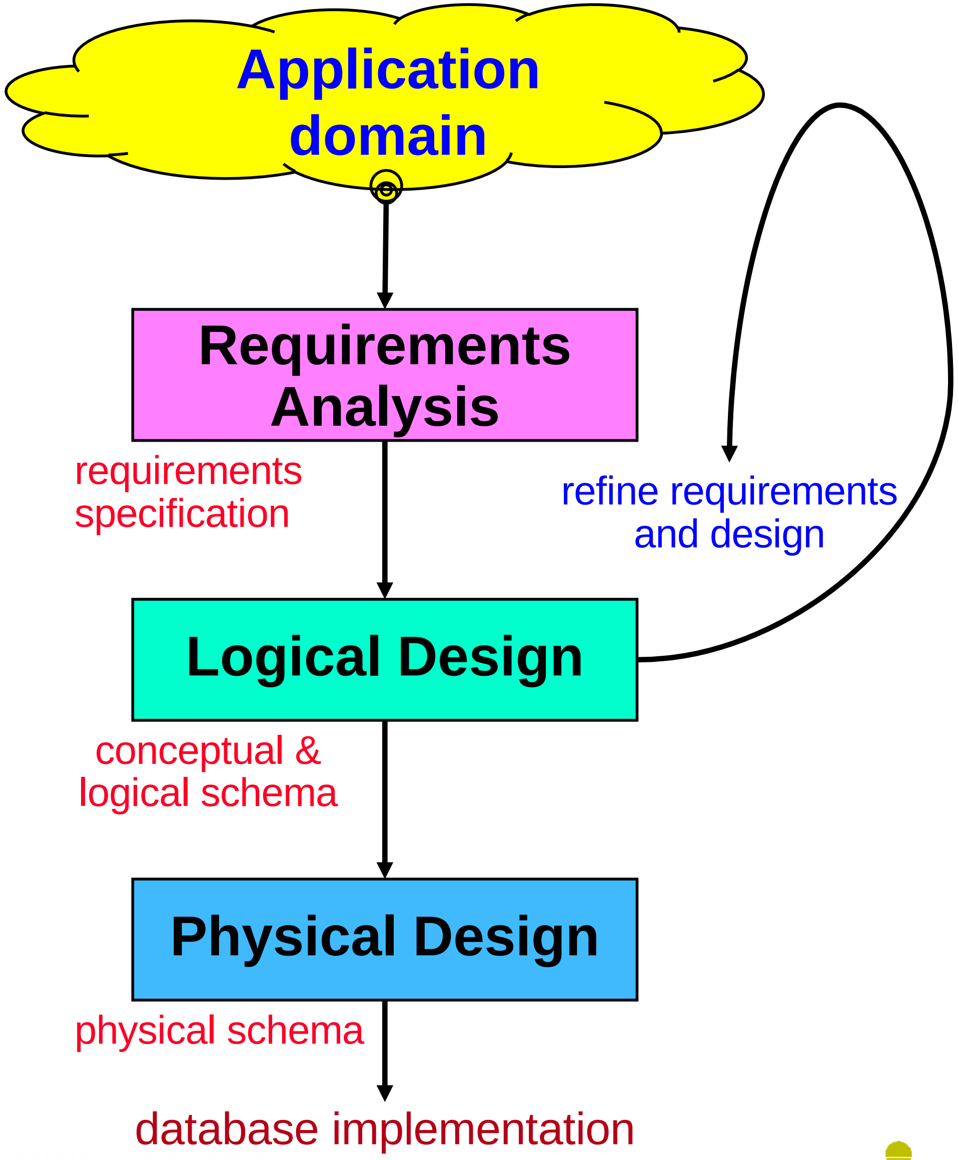
– Entity
– Attribute
– Specialization/Generalization
– Relationship

Entity-Relationship (E-R) Model — Constraints

– Attribute — Domain, Key
– Specialization/Generalization — Coverage
– Relationship — Cardinality, Participation, Exclusion

Analyzing Application Requirements / Making Design Choices

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.).

DATABASE DESIGN PROCESS (CONT'D)

Requirements Analysis produces a requirements specification

- Requirements analysis understands the application domain and describes the data required for processing.

Logical Design produces a conceptual schema and a logical schema

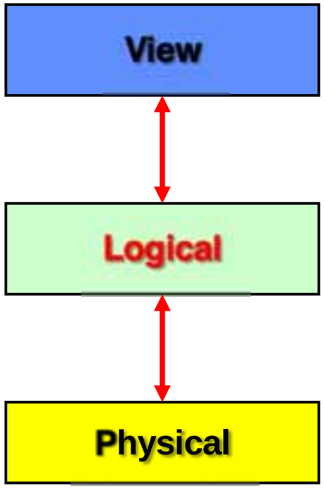
- Logical design describes how the data requirements are represented in the database and often proceeds in two phases producing two schemas.
  a) The conceptual schema describes the requirements for a database using a DBMS independent data model (e.g., the E-R model).
  b) The logical schema describes the database using the data definition language (DDL) of the target DBMS (e.g., SQL DDL).

Physical Design produces a physical schema

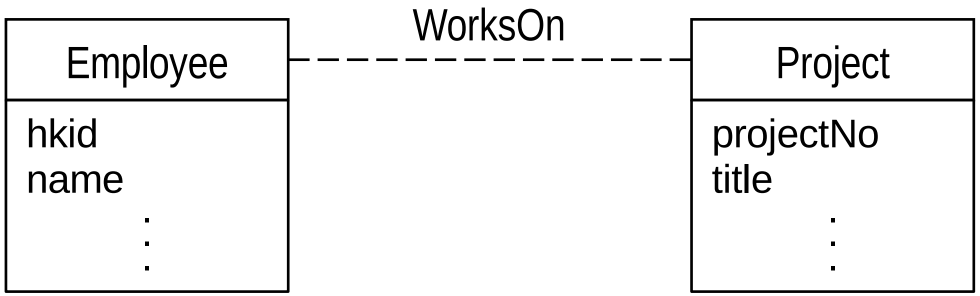
- Physical design describes how the logical schema is stored on the storage media (e.g., data types, keys, indexing options and other parameters).

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. entities.
  2. attributes.
  3. relationships (among entities).



Why E-R model?

- expressiveness
- user communication
- DBMS independent

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

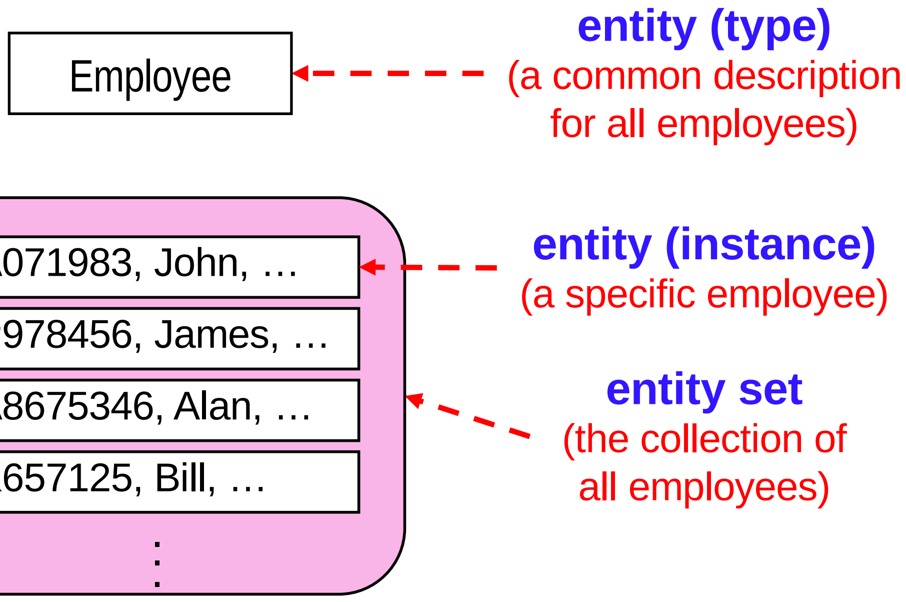
E-R MODEL: *ENTITY*

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:



Employee

entity (type)
(a common description for all employees)

A071983, John, …
P978456, James, …
A8675346, Alan, …
K657125, Bill, …
⋮

entity (instance)
(a specific employee)

entity set
(the collection of all employees)

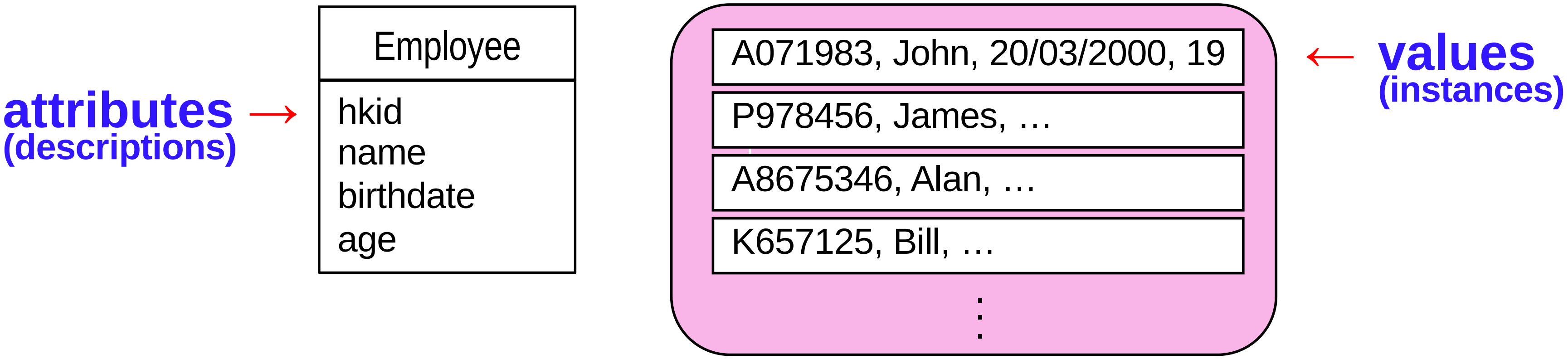
An entity instance

– has identity.
- ➢ It can be distinguished from other entity instances.

– represents some real world thing.
- ➢ It has meaning in the application domain.

E-R MODEL: ATTRIBUTE

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



attributes (descriptions) →

Employee

hkid
name
birthdate
age

A071983, John, 20/03/2000, 19
P978456, James, …
A8675346, Alan, …
K657125, Bill, …
⋮

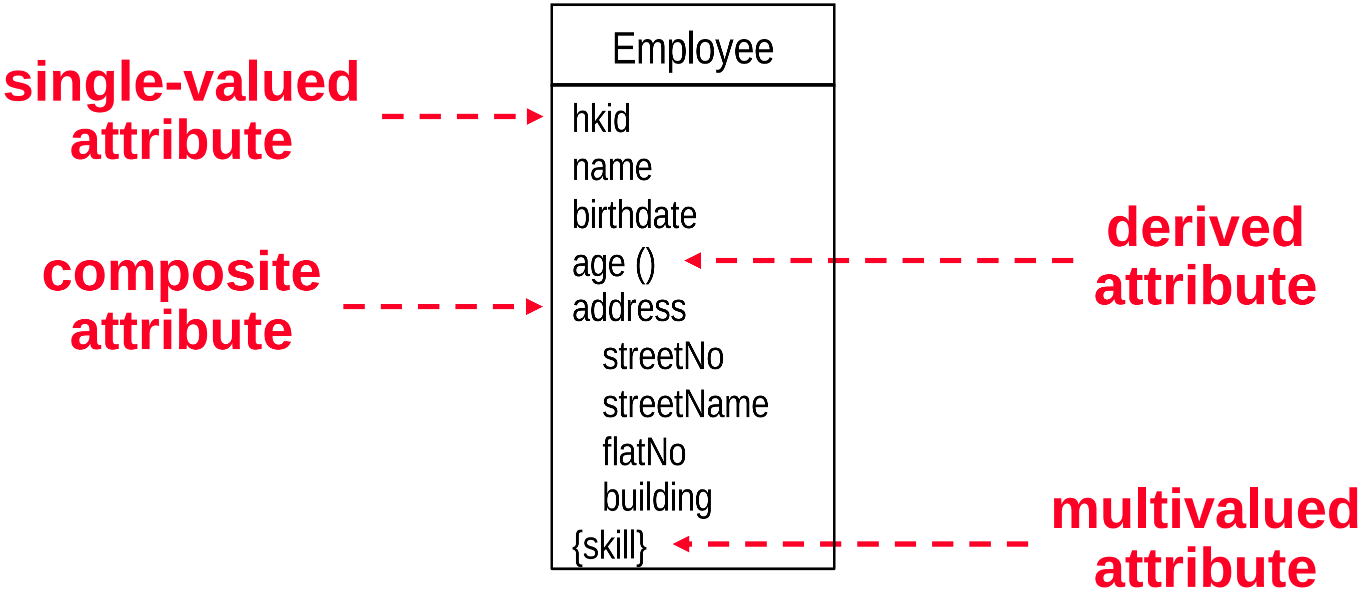
← values (instances)

- 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).

E-R MODEL: ATTRIBUTE—TYPES AND NOTATION



Employee

hkid
name
birthdate
age ()
address
streetNo
streetName
flatNo
building
{skill}

single-valued attribute

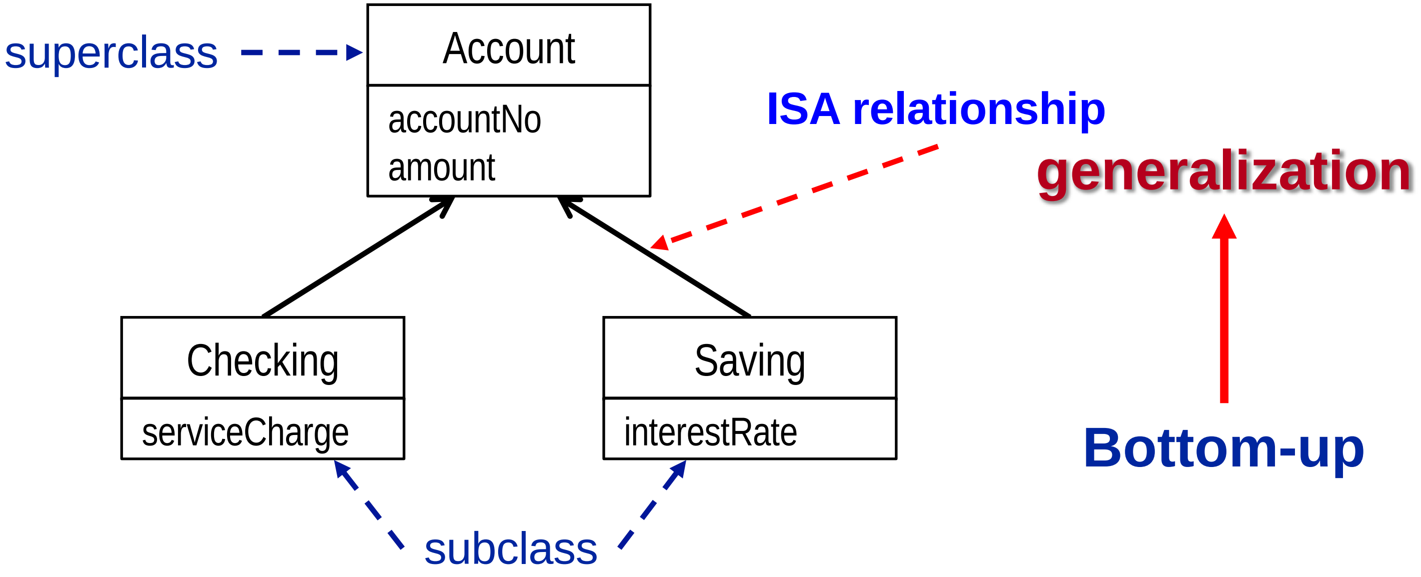
composite attribute

derived attribute

multivalued attribute

E-R MODEL: GENERALIZATION/SPECIALIZATION

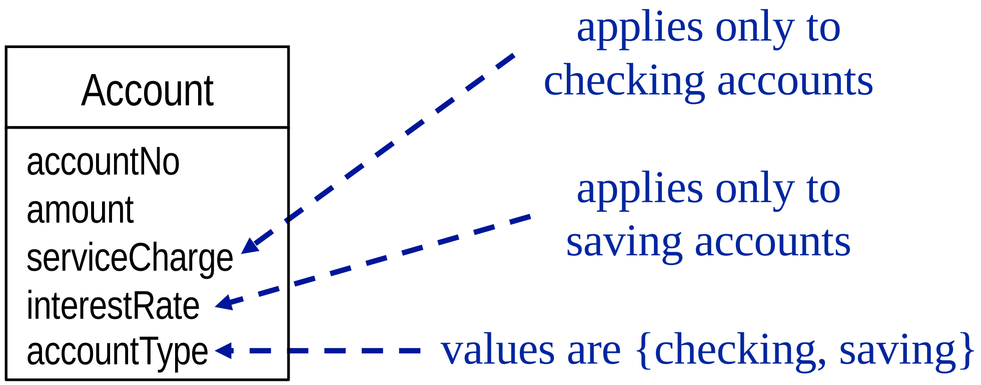
Generalization/specialization **is a relationship between the same kind of entities playing different roles.**



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

E-R MODEL: GENERALIZATION/SPECIALIZATION (CONT'D)

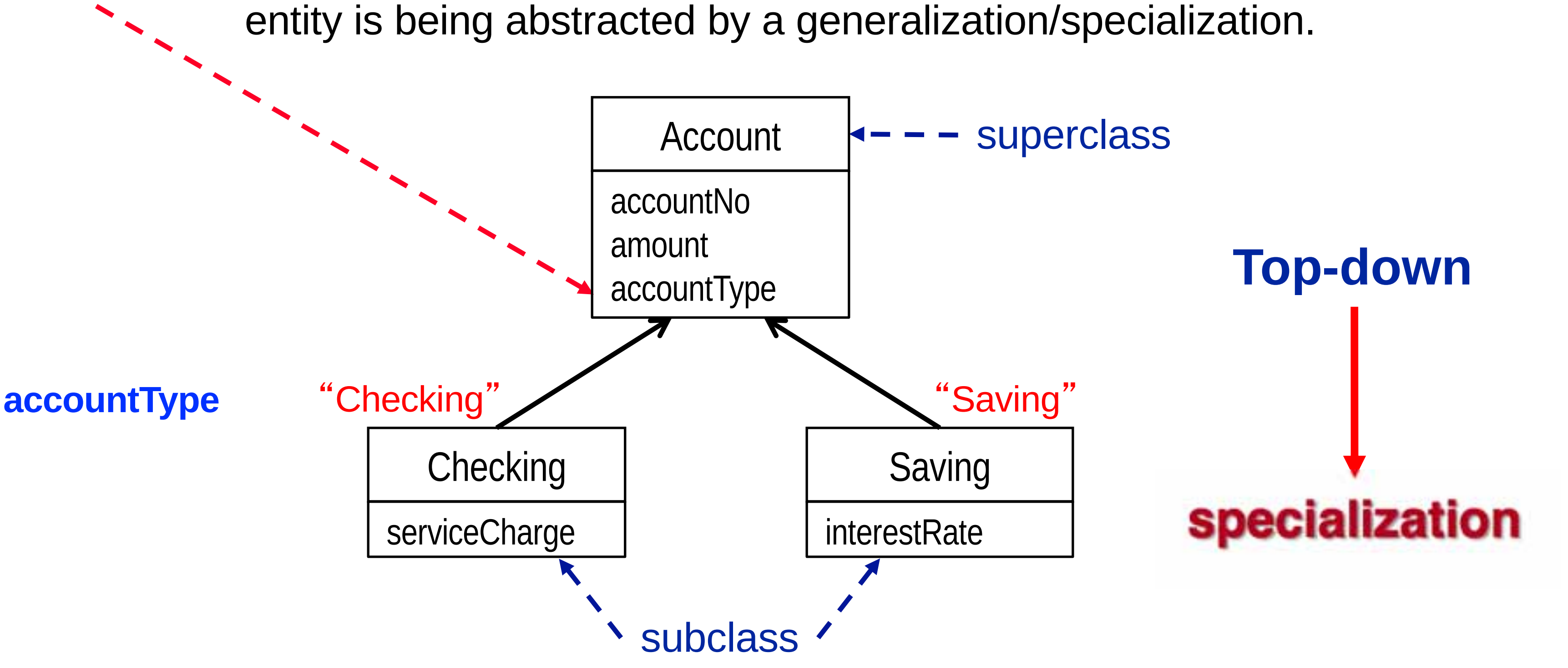
Can also be applied top-down (attribute-defined).


Account
accountNo
amount
serviceCharge
interestRate
accountType
applies only to checking accounts
applies only to saving accounts
values are {checking, saving}

E-R MODEL: 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.

E-R MODEL: GENERALIZATION 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.**

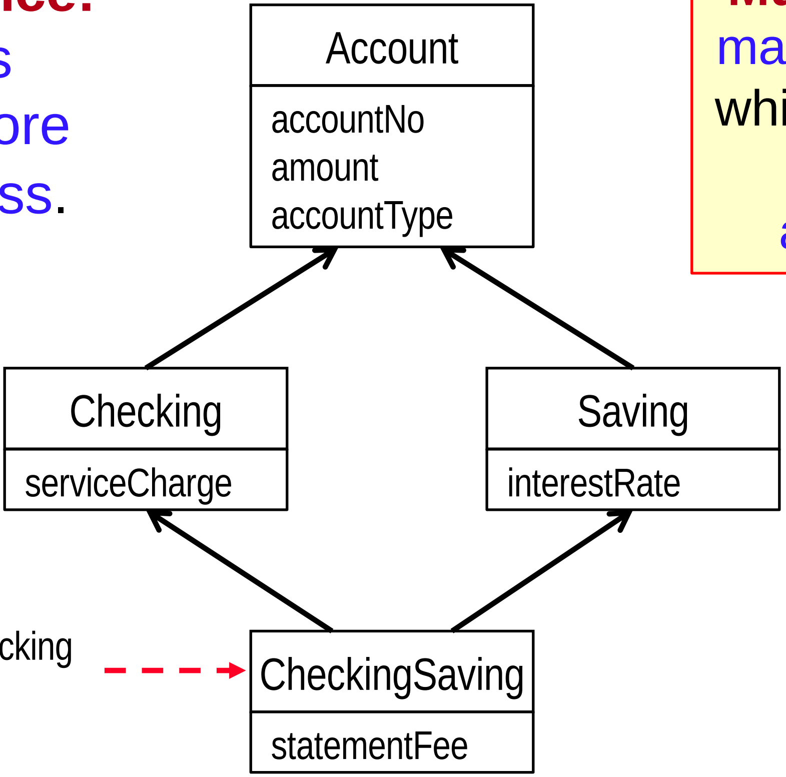
- A subclass may add new properties (attributes, relationships).

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

E-R MODEL: GENERALIZATION
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.

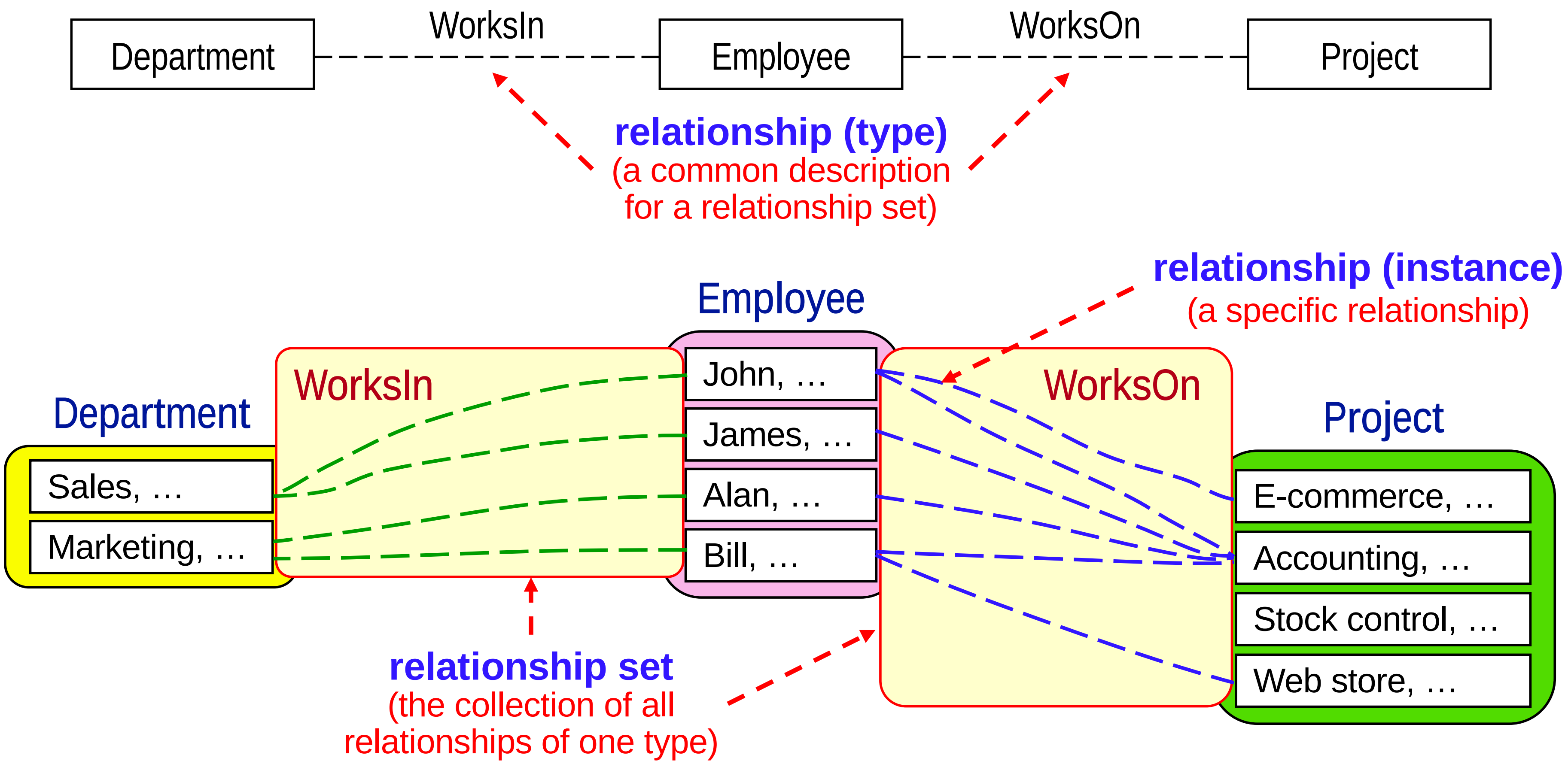


Inherits from *both* Checking and Saving entities.

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

E-R MODEL: RELATIONSHIP

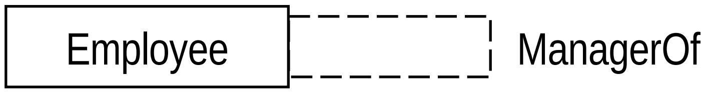
A *relationship (type)* is a description of a set of relationships with common properties and semantics.



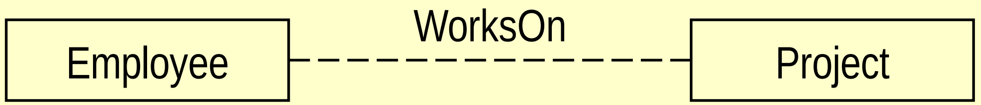
RELATIONSHIP: DEGREE

- The number of entity types that participate in a relationship type.

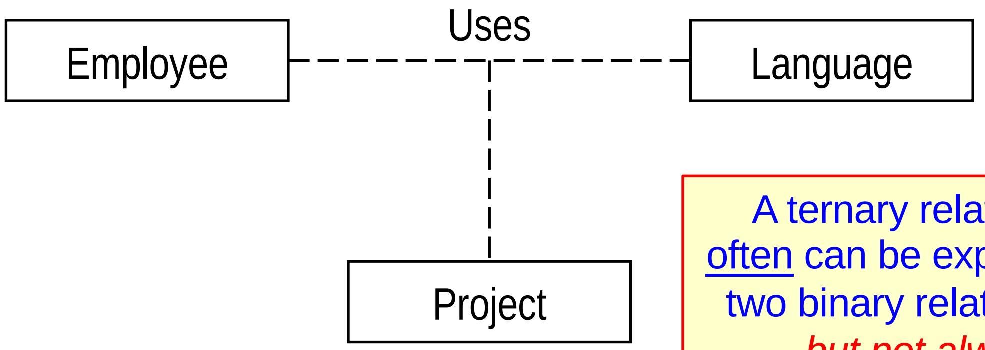
unary (reflexive) –
relates *one* entity (to itself)



binary –
relates *two* entities



ternary –
relates *three* entities



Higher degrees are extremely rare!

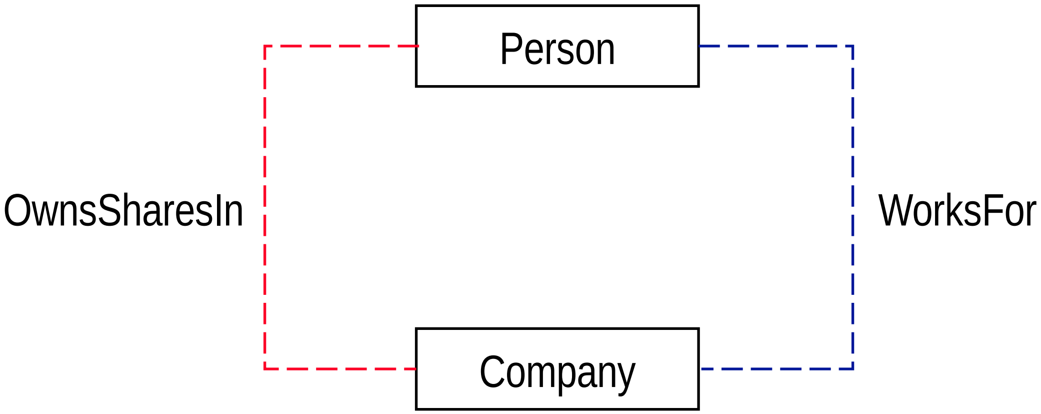
A ternary relationship often can be expressed as two binary relationships, *but not always*.

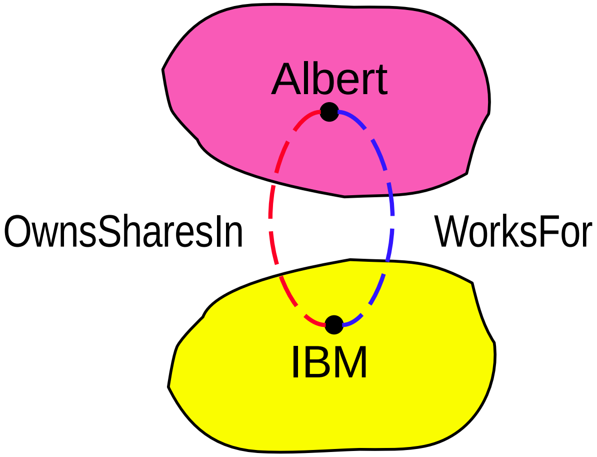
In practice, the vast majority of relationships are binary.

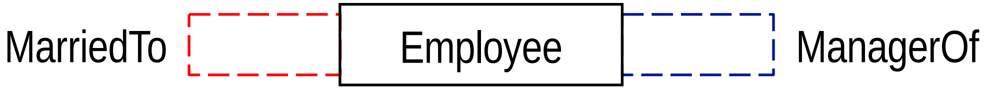
(We will use only unary or binary relationships in this course.)

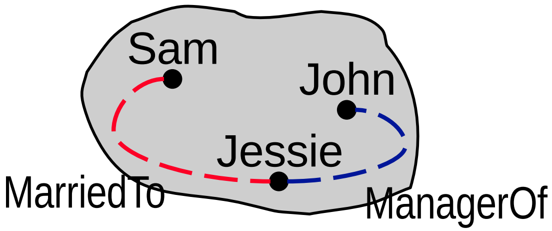
E-R MODEL: RELATIONSHIP EXAMPLES

There can be several relationships between entities.



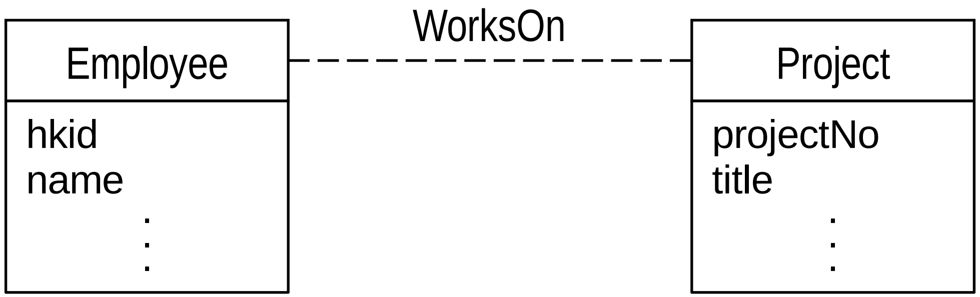


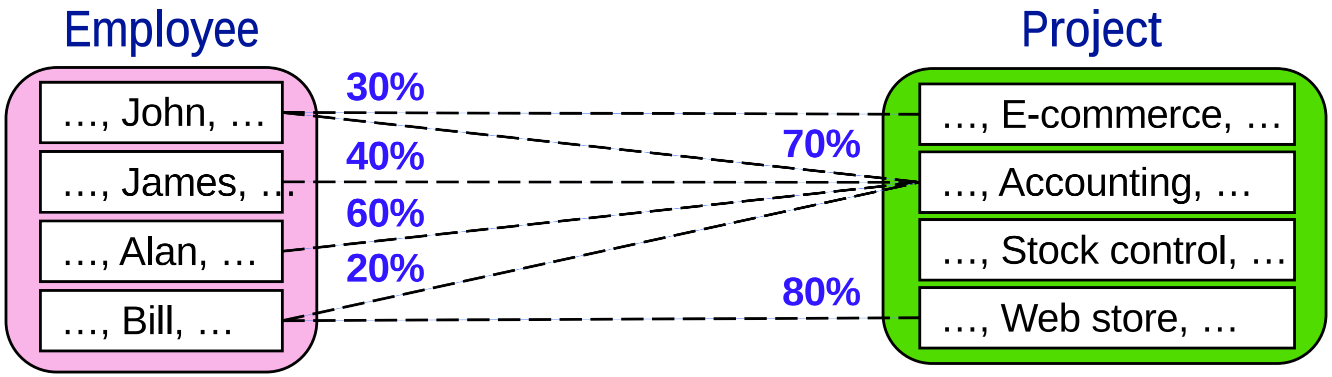




RELATIONSHIP: RELATIONSHIP ATTRIBUTE

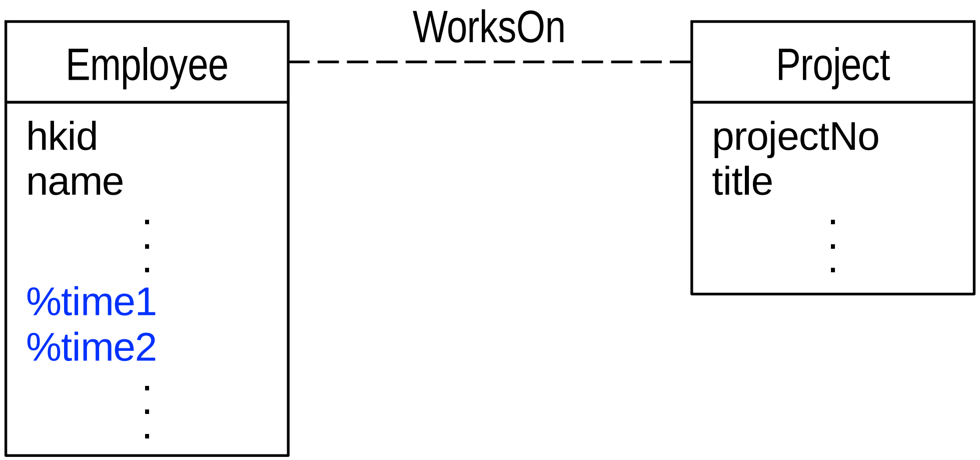
- We want to represent the percentage time worked on a project.

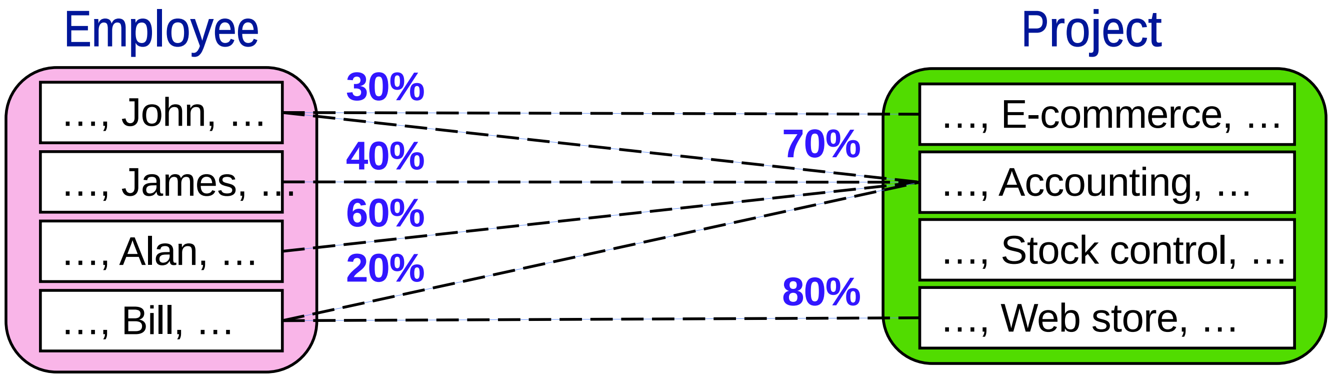




RELATIONSHIP: RELATIONSHIP ATTRIBUTE (cont'd)

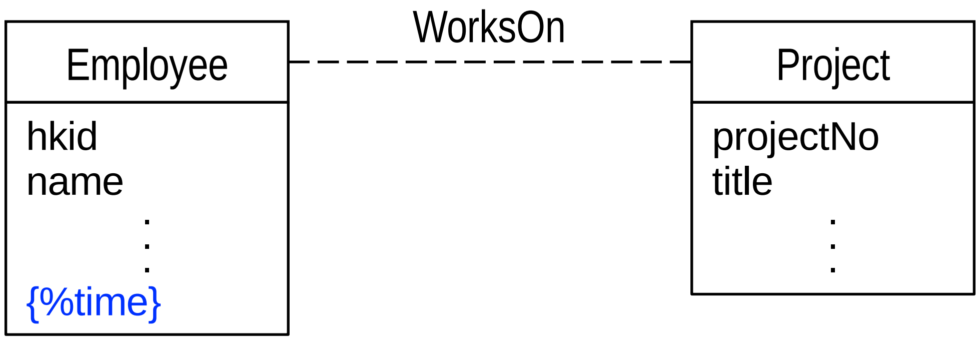
Option 1: Use many attributes (e.g., in Employee). **Is this OK?**

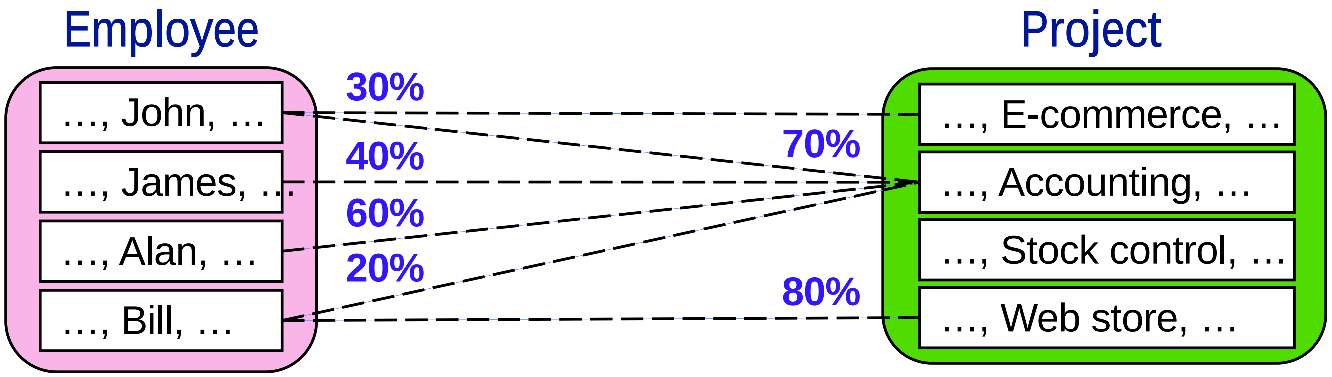




RELATIONSHIP: RELATIONSHIP ATTRIBUTE (cont'd)

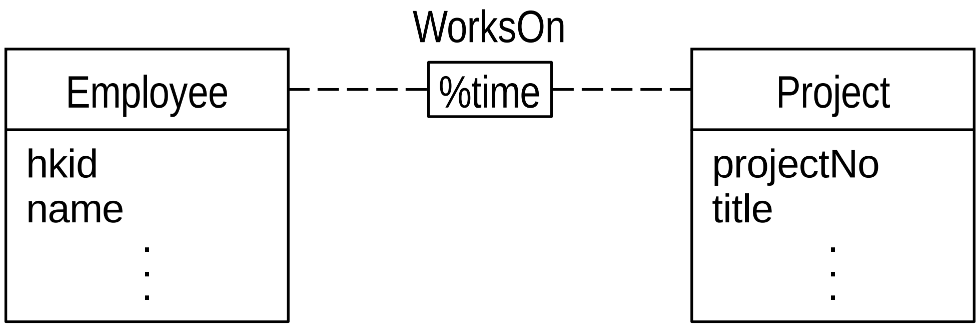
Option 2: Use a multivalued attribute (e.g., in Employee). **Is this OK?**

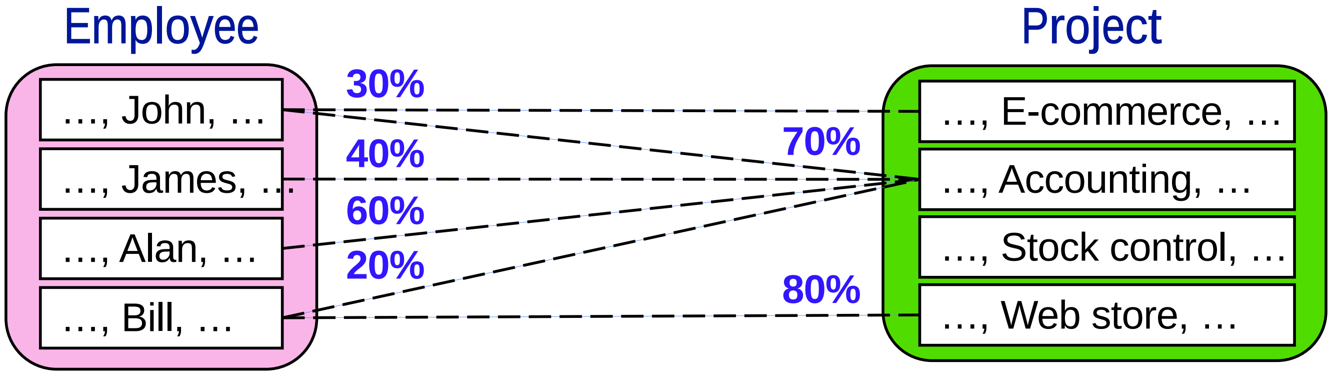




RELATIONSHIP: RELATIONSHIP ATTRIBUTE (cont'd)

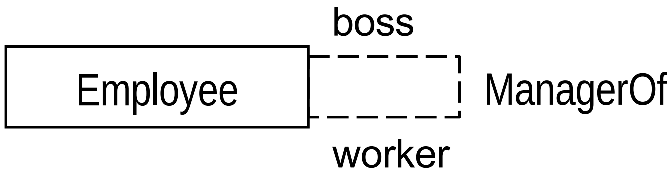
Option 3: Allow relationships to have attributes. **Is this OK?**

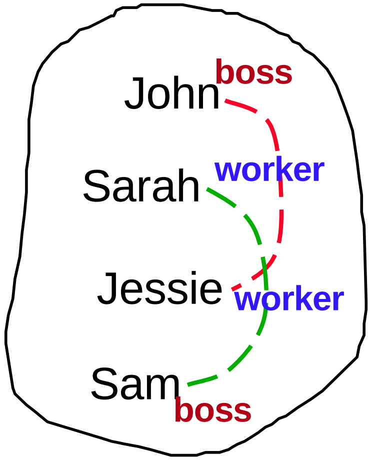




RELATIONSHIP: ROLE NAME

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).