+

COMP3311 Database Management Systems

Spring 2022

The ER Model

Prof Xiaofang Zhou

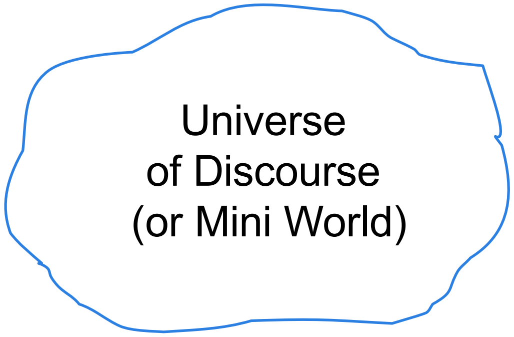
香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

+ In This Lecture…

- Why conceptual modelling is important for designing a successful database application?
  - Without conceptual models, it is very difficult to communicate database designs to (non-technical) users - this lack of communication may result in user’s data requirements being missed or incorrect requirements being captured
- How to use the Entity Relationship (ER) Model, one of the most widely used method for conceptual modelling?

+ Phases of Database Design

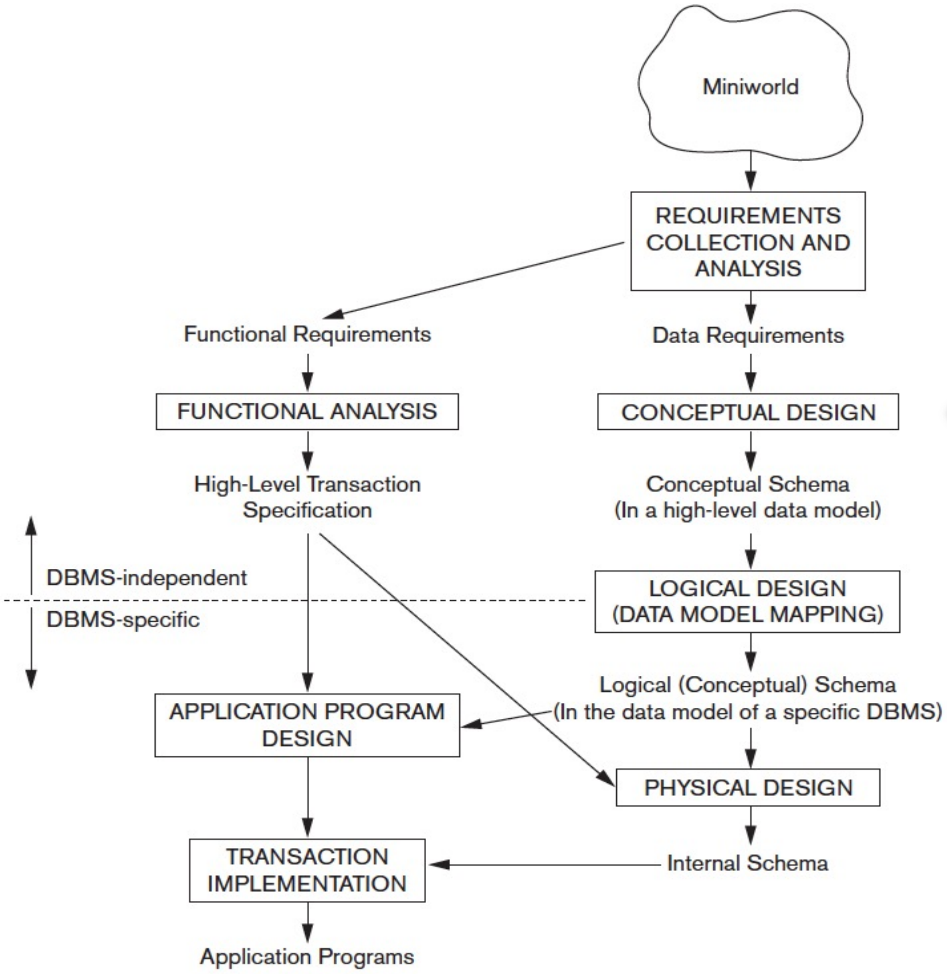
- First step of the design process is to identify the “Universe of Discourse” (UoD)
- The database to be built will not model everything in the world, but rather some “mini-world” or “Universe of Discourse”.
- The UoD is the relevant portion of the real world to be modeled by the database, i.e a mini world



Universe of Discourse (or Mini World)

…close-world assumption

+ Phases of Database Design


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

focus of this lecture

… conceptual (Data) design and functional (Program) design are closely related, should be done in parallel

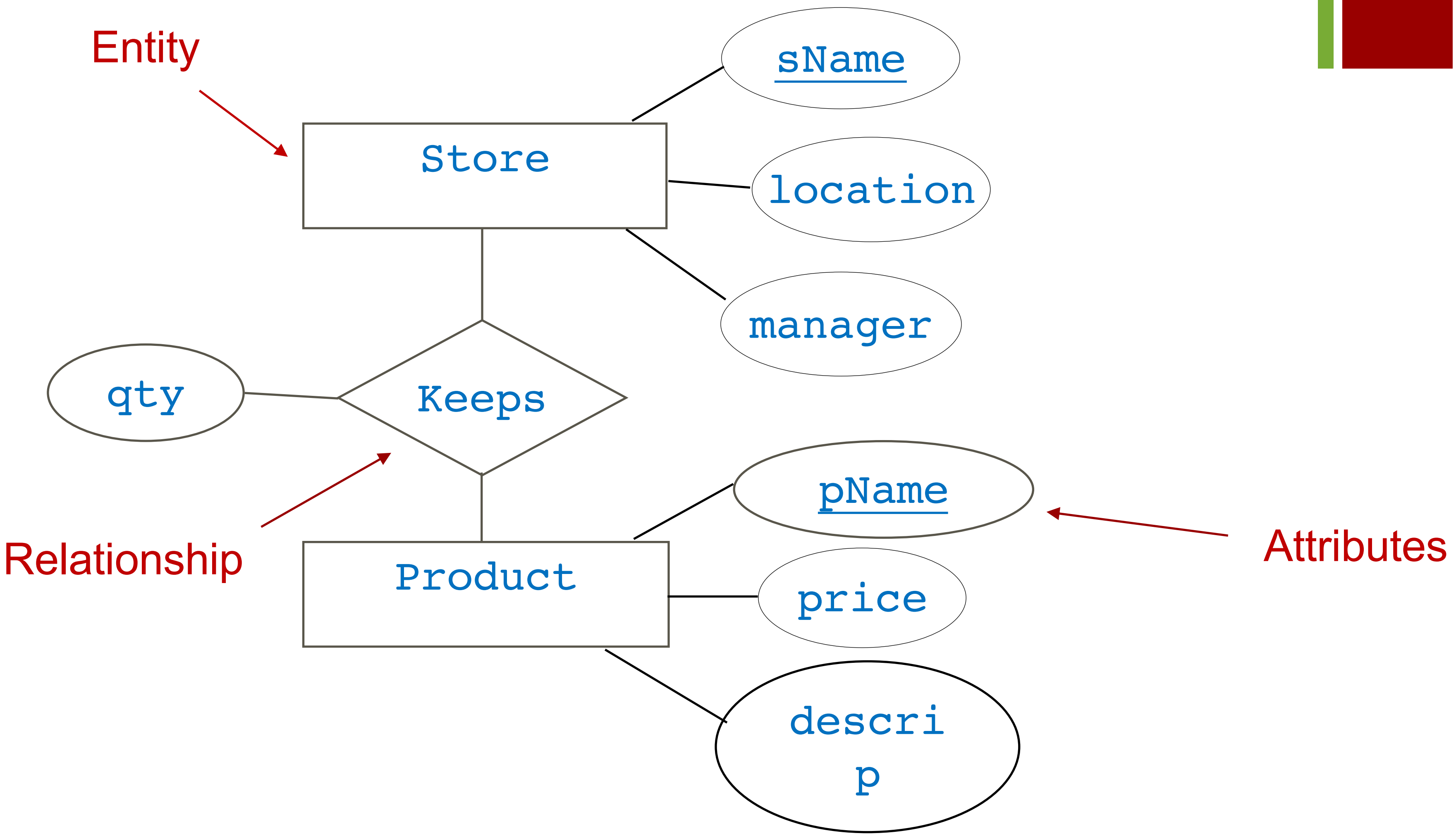
+ The ER Model Basics

- Entities
  - Entities, Entity Sets and Entity Types
  - Attributes, Keys and Value Sets
- Relationships
  - Relationship Types and Sets
  - Relationship Degree
  - Roles and Recursive Relationships
  - Relationship Constraints
  - Attributes of Relationship Types

+ The Entity-Relationship Model

- ER is a model for the logical level
  - It describes the structure of the database at a high abstraction level
- A database can be modeled as
  - A collection of entities, and
  - Relationships among entities

+ An ER Diagram Example



+ Entity

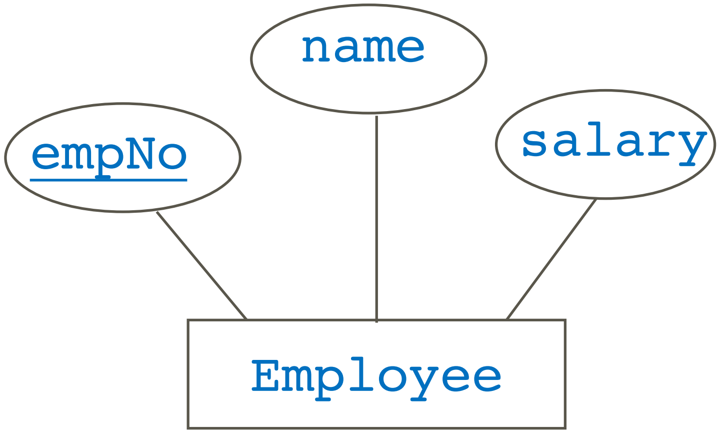
- An entity is a real-world “thing” or “object” distinguishable from other objects
  - E.g., a student, car, job, course, building
  - Color, age, weight, grade are not
- An entity is described using a set of attributes
- The same entity may have different prominence in different UoDs
  - In a Company database, an employee’s car is of lesser importance
  - In a Department of Transportation’s registration database, cars may be the most important concept
  - In both cases, cars will be represented as entities; but with different levels of information captured

+ Entity Type and Entity Set

- Each entity type is described by its name and attributes (e.g. an employee)

- A collection of similar entities at a given point in time is called and entity set (e.g. all employees)
  - All entities in an entity set have the same set of attributes

- Entity type and entity set are customarily referred to by the same name



...the entity type describes the "Schema" or "Intension" for a set of entities
...the entity set is also called "Extension" of an entity type

+ Relationship Types and Sets

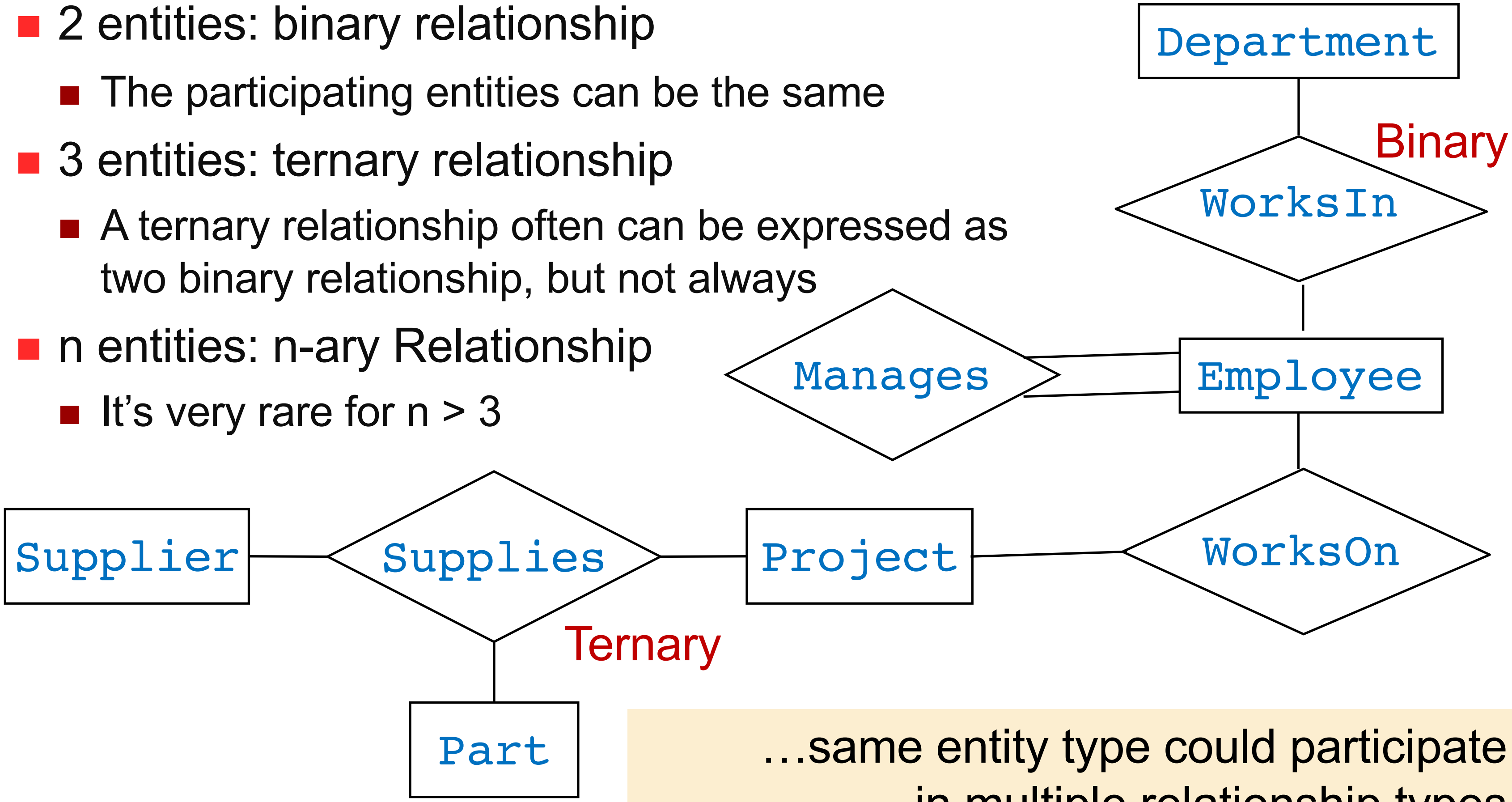
- A relationship is an association among two or more entities
  - E.g., John works in Pharmacy Department
- A relationship type defines the relationship, and a relationship set represents a set of relationship instances
  - Relationship type and corresponding set are customarily referred to by the same name

... subject-verb-object

… an entity type is typically represented as a singular form of noun, and a relation type is often represented as a verb

+ Relationship Degree

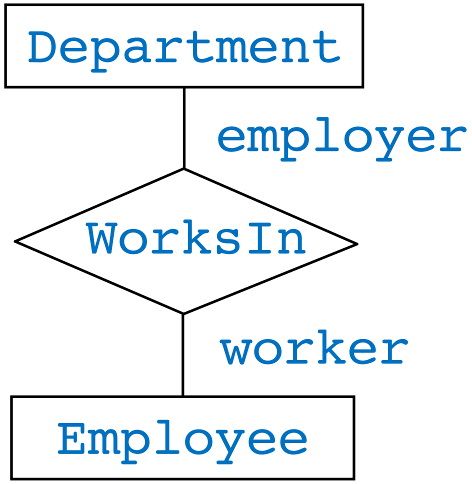
- The degree of a relationship type is the number of participating entity types
  - 2 entities: binary relationship
    - The participating entities can be the same
  - 3 entities: ternary relationship
    - A ternary relationship often can be expressed as two binary relationship, but not always
  - n entities: n-ary Relationship
    - It's very rare for n > 3

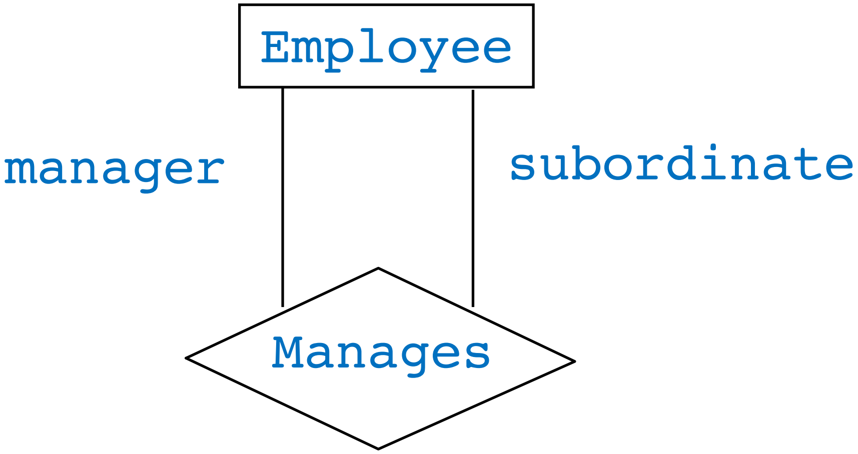


…same entity type could participate in multiple relationship types

+ Entity Roles

- Each entity type that participates in a relationship type plays a particular role in the relationship type
- The role name signifies the role that a participating entity from the entity type plays in each relationship instance, i.e. it explains what the relationship means



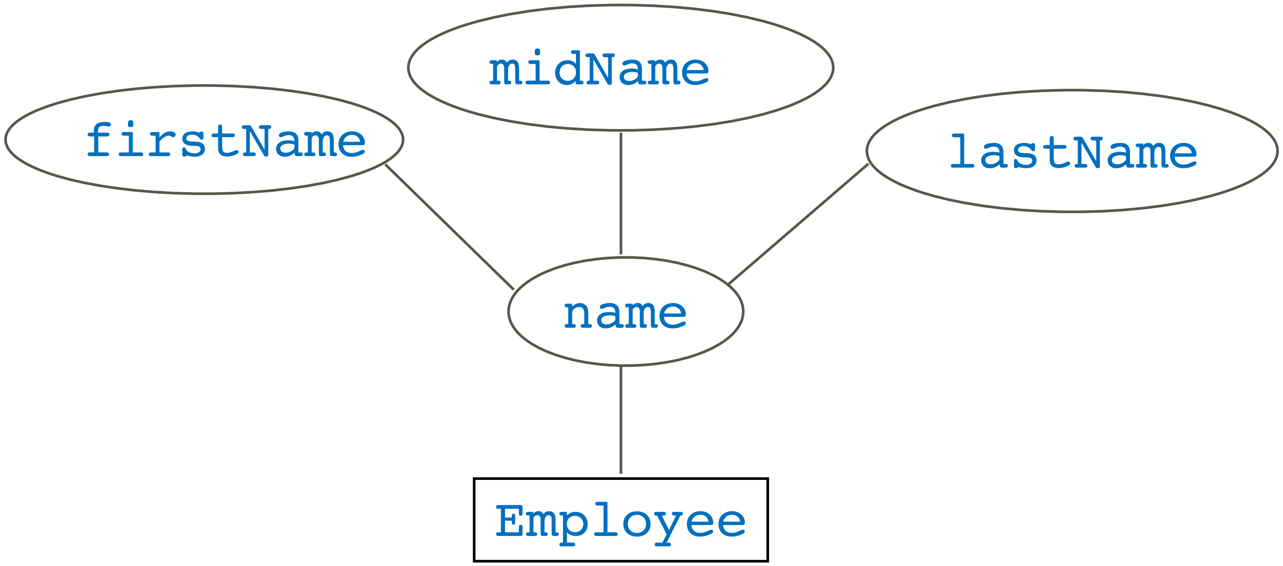


+ Attributes

- Properties of an entity or a relationship
  - `name`, `address`, `weight`, `height` are properties of the `Person` entity
  - `dateOfMarriage` is a property of the `Marriage` relationship

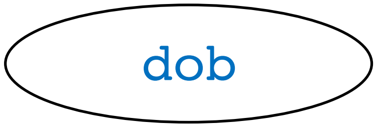
+ Simple vs. Composite Attributes

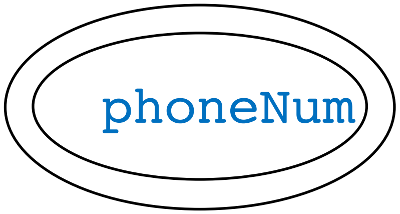
- Composite attributes can be divided into smaller parts which represent simple attributes with independent meaning



+ Single Vs. Multivalued Attributes

- Simple attributes can either be single-valued or multi-valued





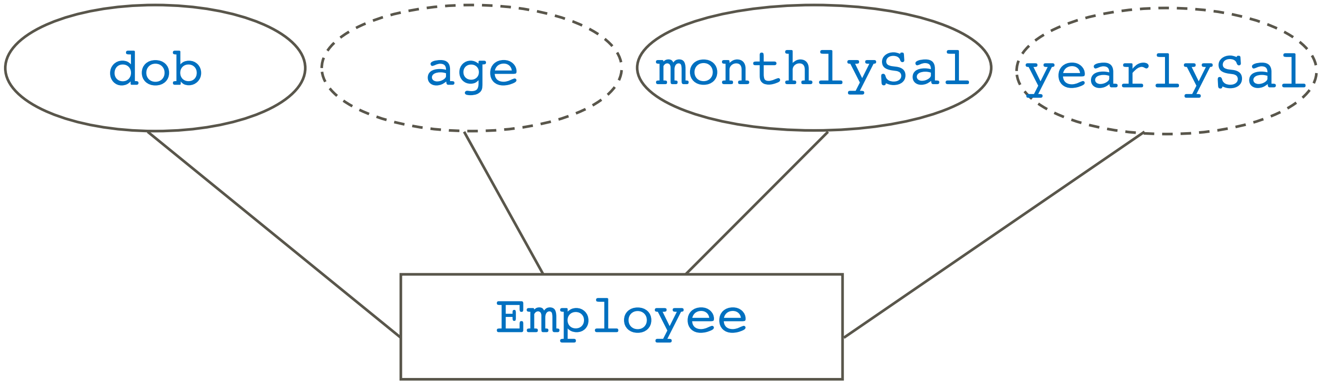
Single value attribute: only one value is acceptable (e.g., one person’s date if birth)

Multivalued value attribute: multiple values can be acceptable (e.g., phone numbers, email addresses)

... the relational model doesn’t support multi-valued attributes, but we are doing conceptual modeling now

+ Derived Vs. Stored Attributes

- Some attribute values can be derived from related attribute values:
  - `age` = `date` - `dob`
  - `yearlySal` = 12 * `monthlySal`

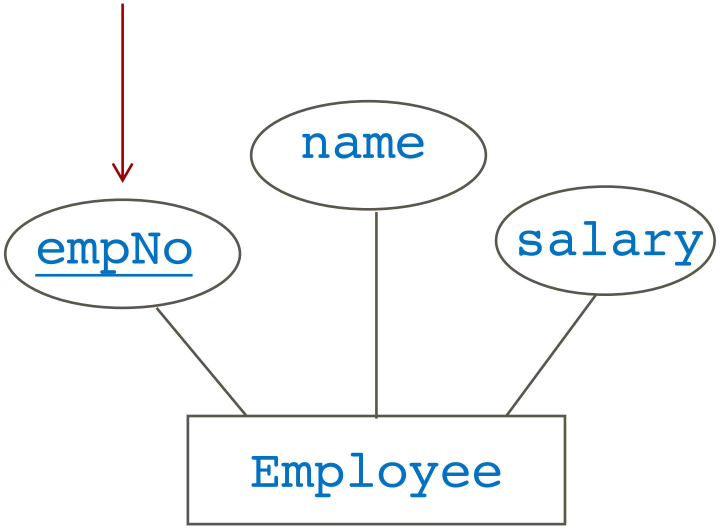


... anything that can be derived is also called redundant
…what problems do you see for redundancy?

+ Key Attributes

- A key is a set of attributes whose values are distinct for every entity in the entity set
  - Composite key: a key with multiple attributes
  - Multiple keys are possible
  - A key with any other attribute is also a key (called a superkey)
  - A candidate key is a minimal set of attributes to be a key
  - One candidate key is designated as the primary key

A key attribute has its name underlined inside the oval



…a key must hold for every possible extension of the entity type
…sometimes artificial keys maybe created

+ Value Sets of Attributes

- Value sets specify the set of values that may be assigned to a particular attribute of an entity
  - `employeeAge`: integers between 17 & 65
  - `vehicleRegistrationNum`: string of 3 alphabets followed by 3 integers
- Value sets map to relational domains
- Value sets are <u>not</u> displayed on the ER diagram

+ Null Valued Attributes

- A particular entity may not have an applicable value for an attribute
  - `tertiaryDegree`: not applicable for a person with no university education
  - `homePhone`: not known if it exists
  - `height`: not known at present time
- A null value may have many different meanings
  - Not applicable
  - Unknown
  - Missing...

+ Exercise

Every student has a unique id, name (composed of title, first name, middle initial and last name), email, address (composed of number, street name, suburb and postcode), and one or more phone numbers.

+ Relationship Constraints

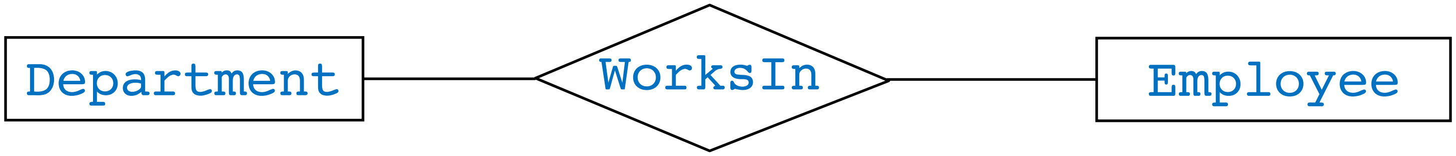
- Constraints on relationships are determined by the UoD these relationships are describing

- Constraints on the relationship type limit the possible combination of entities that may participate in the corresponding relationship set
  - Cardinality constraints
  - Participation constraints

… these are “structural constraints” which the ER model can capture

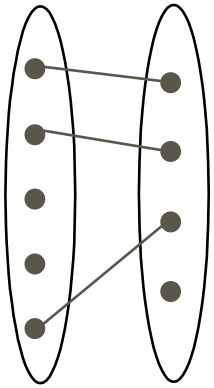
+ Cardinality Ratio

- The cardinality ratio for a binary relationship specifies the number of relationship instances that an entity can participate in
  - WorksIn is a binary relationship
  - Participating entities: Department and Employee
  - What about the semantics that one department can have many employees?
    - I.e., the cardinality ratio is 1 : n

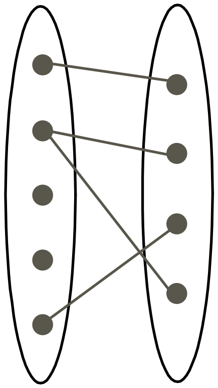


+ Possible Cardinality Ratios

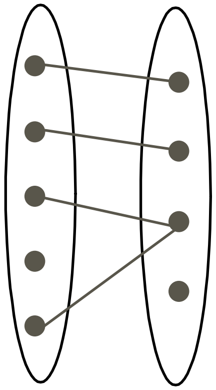
- 1-to-1 (1: 1)
  - Both entities can participate in only one relationship instance
- 1-to-Many (and Many-to-1)
  - One entity can participate in many relationship instances
- Many-to-Many (m: n)
  - Both entities can participate in many relationship instance



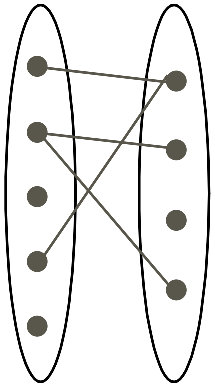
1-to-1



1-to-n



n-to-1



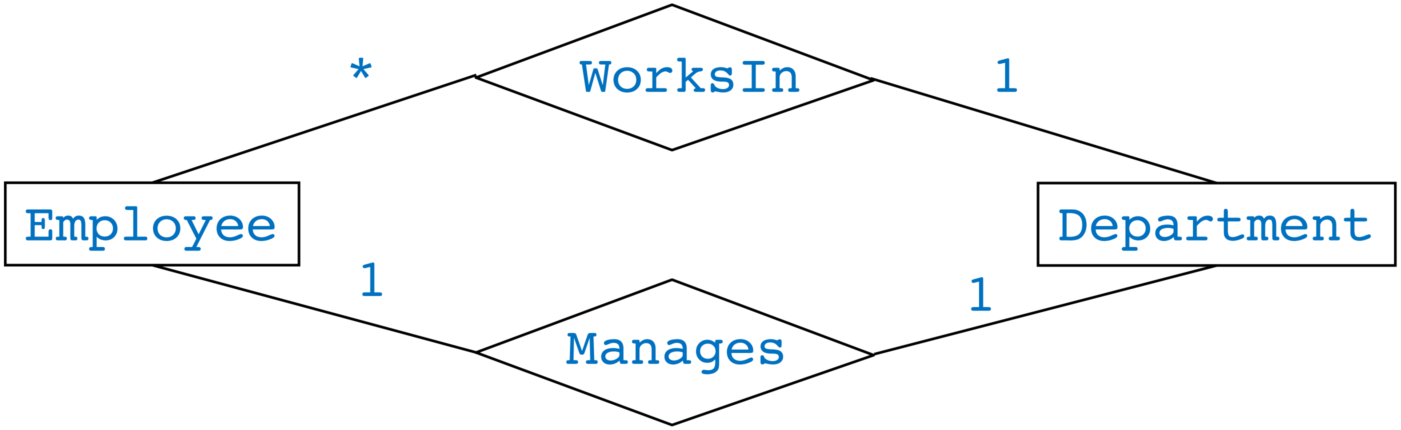
m-to-n

+ Example Cardinality Constraints

- How many Employees can work in a Department?
  - *One employee can work in only one department*
- How many Employees can be employed by a Department?
  - *One department can employ many employees*
- How many managers can a department have?
  - *One department can have only one manager*
- How many departments can an employee manage?
  - *One employee can manage only one department*

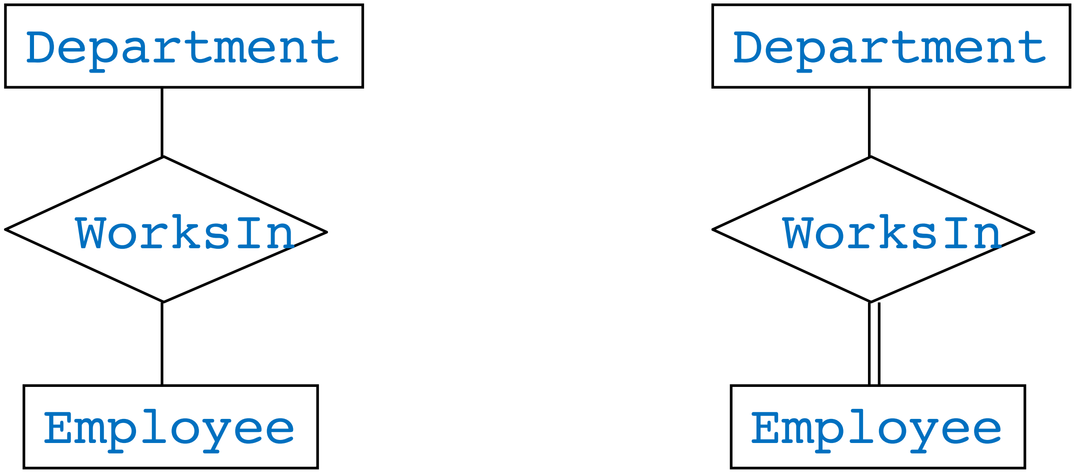
… you should talk to your customers to understand and articulate these constraints, using ER diagram!

+ Representing Cardinality

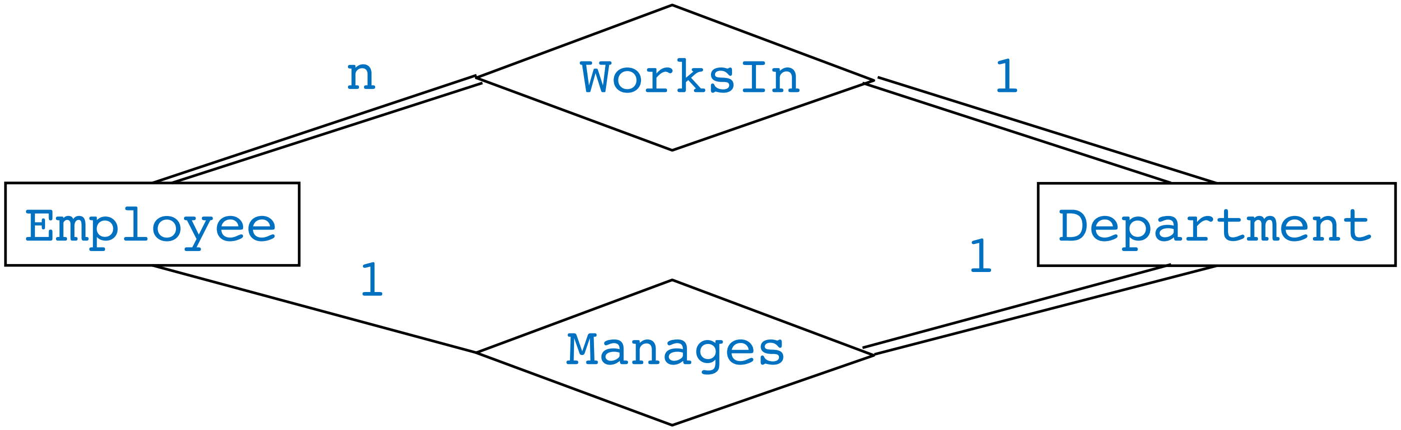


+ Existence Dependency

- It indicates whether the existence of an entity depends on its relationship to another entity via the relationship type
  - Every employee must work for a department
    - This is called total participation, indicated by double line
  - A department may have no employees
    - This is called partial participation

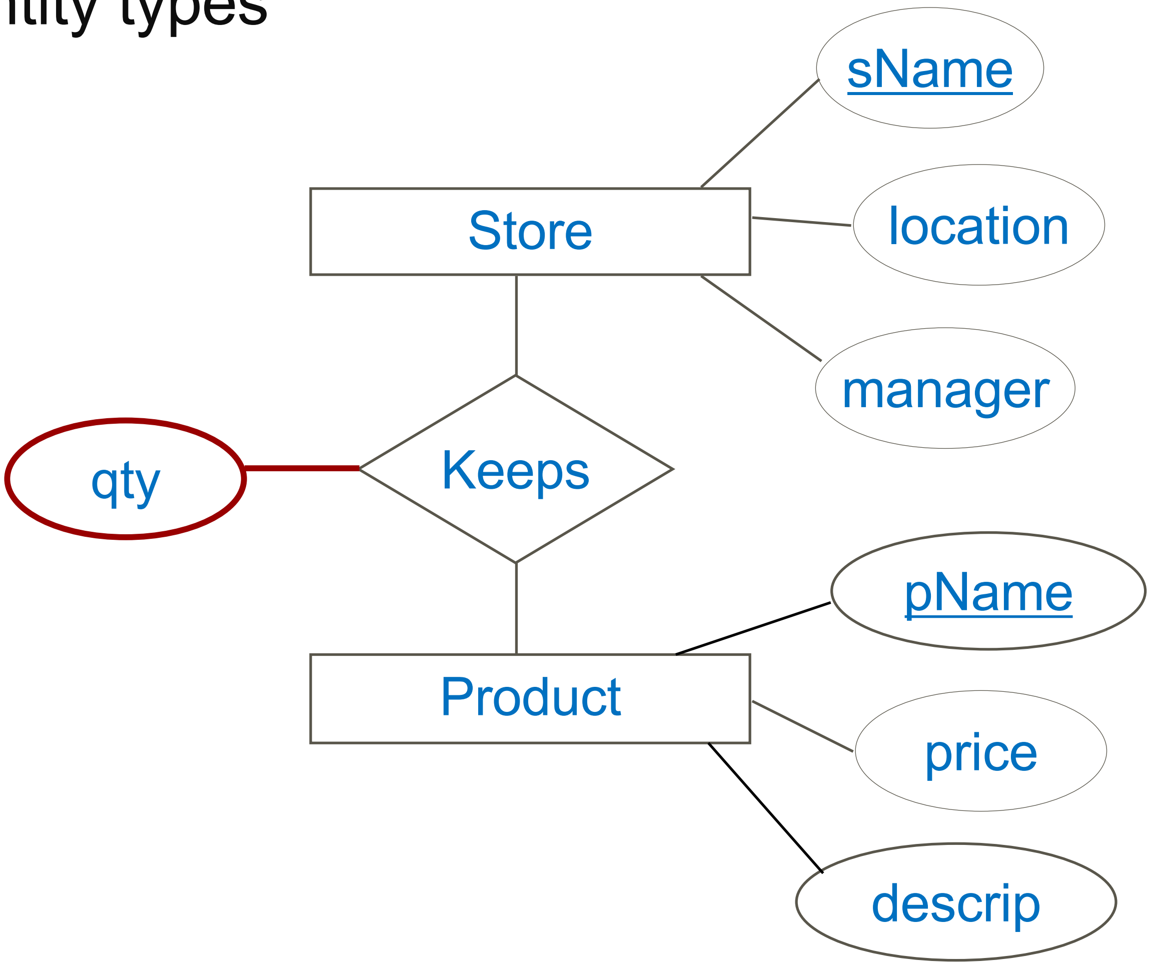


+ What Does This One Tell?



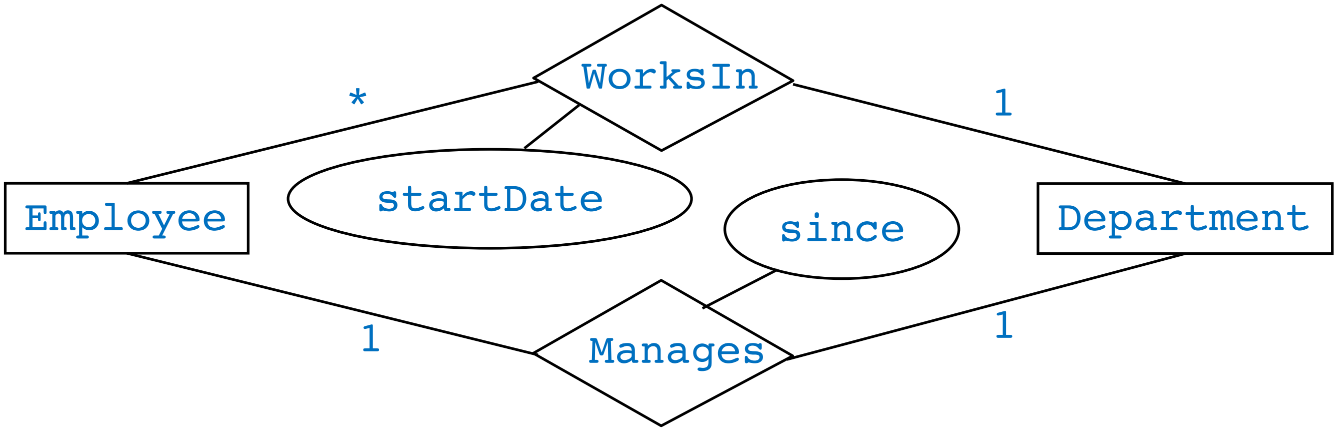
+ Attributes of Relationship Types

- Relationship types can also have attributes just as entity types



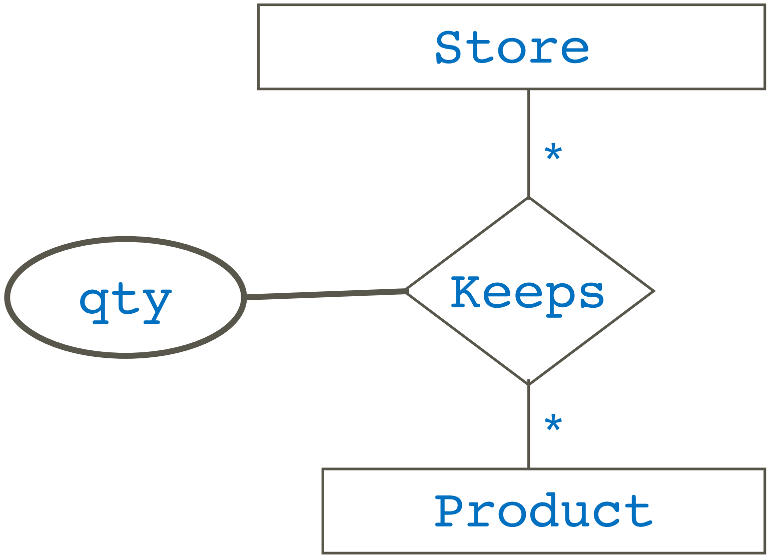
+ Attributes of 1:1 or 1:n Relationship

- Attributes of 1:1 or 1:n relationship types can be migrated to one of the participating entity types
- Examples:
  - `since` of `Manages` can be an attribute of `Employee` (manager) or `Department`
  - `startDate` of `WorksIn` can be an attribute `Employee` (only the n-side of the relationship)



+ Attributes of m:n Relationship

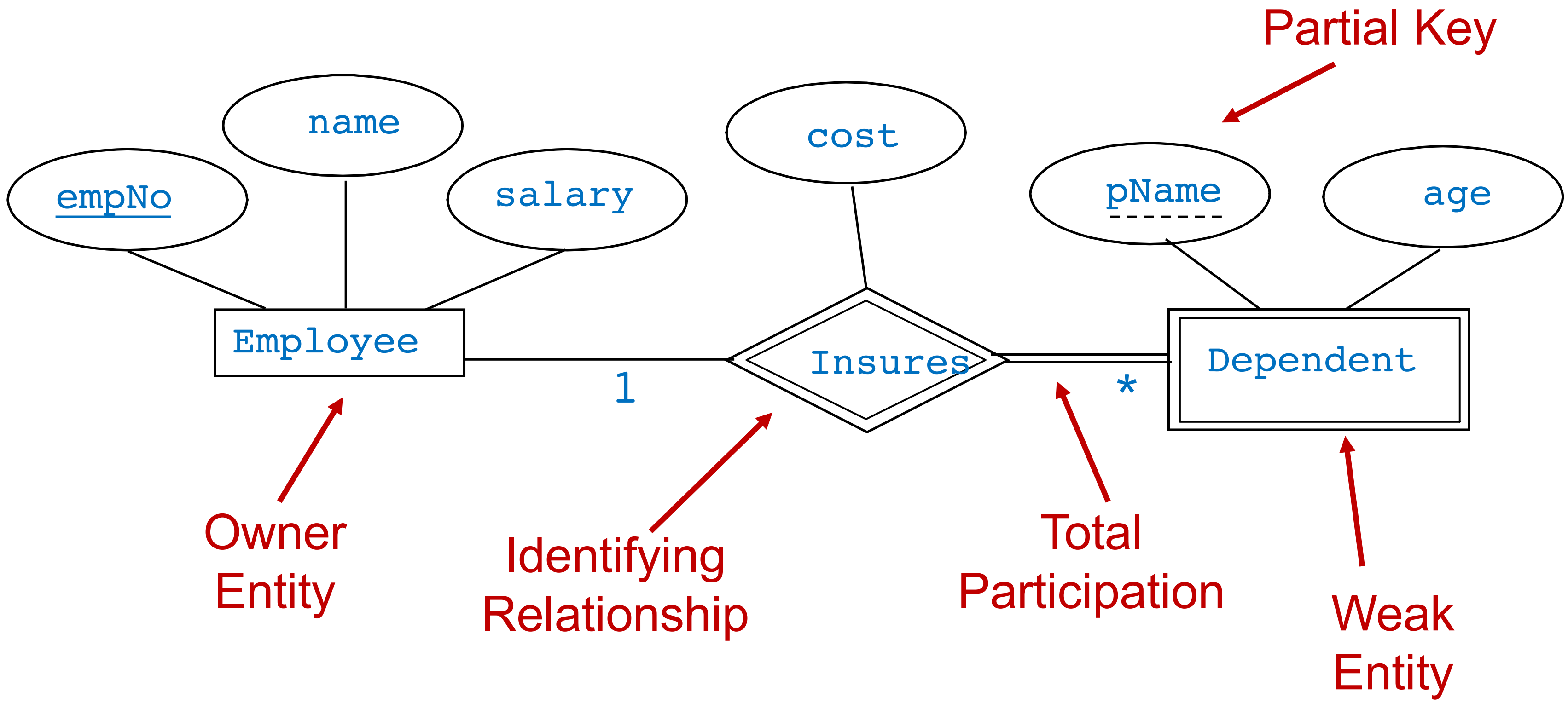
- Attributes of m:n relationship types cannot be migrated to one of the participating entity types
- `qty` of `Keeps` can only be determined by the combination of `Store` and `Product`



+ Weak Entities

- Entity types that do not have key attributes of their own are called weak entities
- A weak entity can be identified uniquely only by considering the primary key of another owner entity
  - They have a partial key
- The relationship type that relates a weak entity to its owner is called the identifying relationship

+ An Weak Entity Example



Partial Key

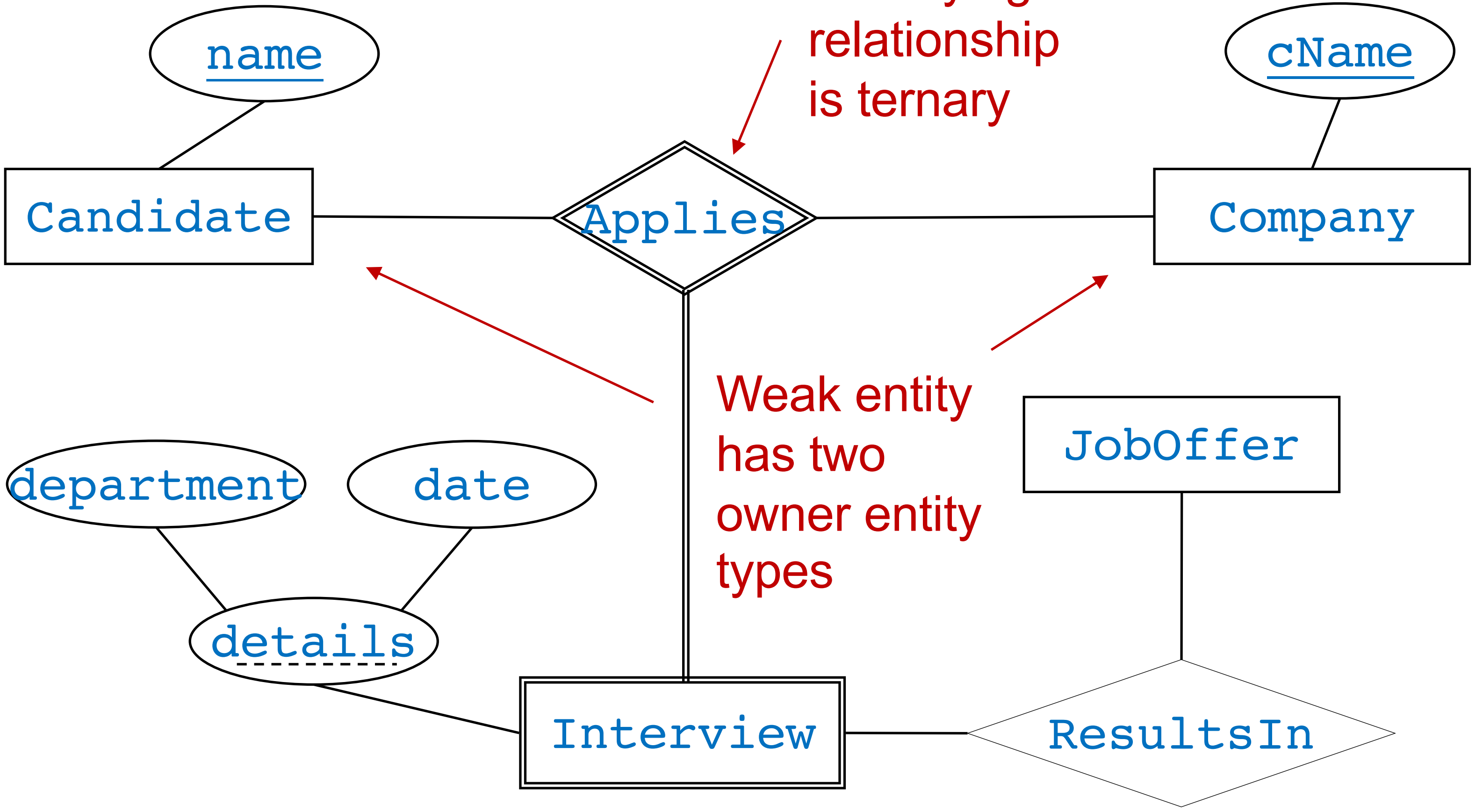
Owner Entity

Identifying Relationship

Total Participation

Weak Entity

+ One More Example

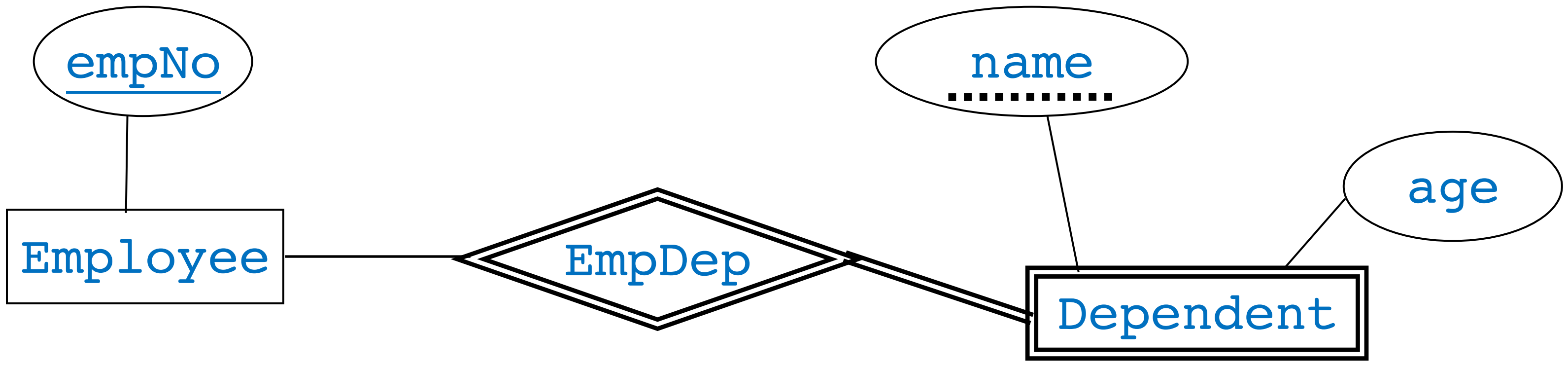


Identifying relationship is ternary

Weak entity has two owner entity types

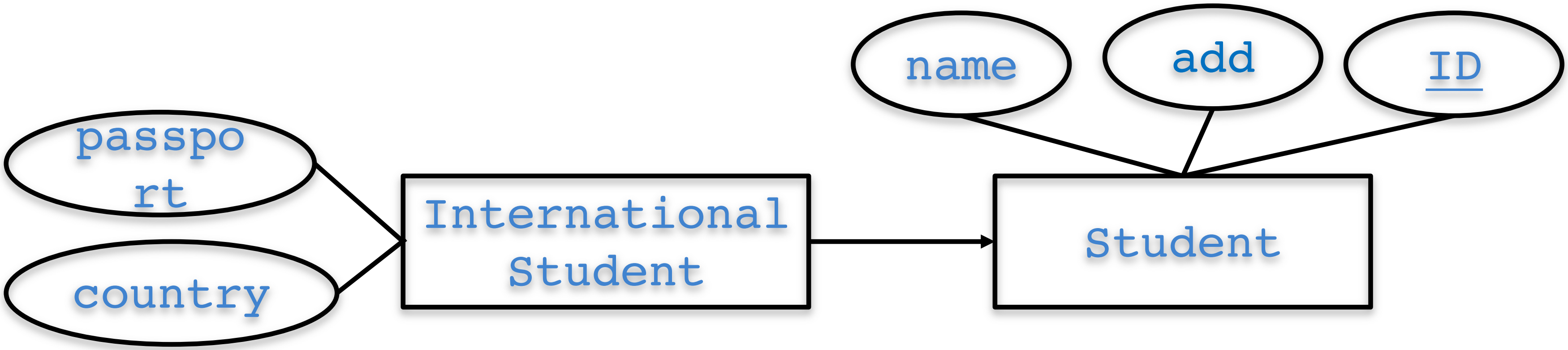
+ Another Example

- A dependent child may not be old enough to have a HKID number
- Even if he/she has a HKID number, the company may not be interested in keeping it in the database



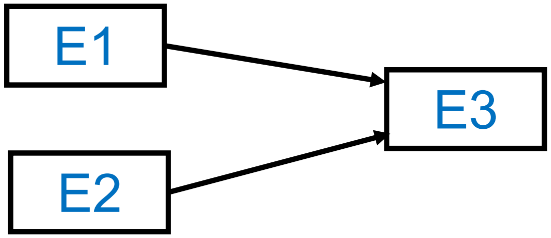
+ Extended ER

- Entity Type is called class in EER
  - Entities in the same class have the same attributes
- Class can be a superclass or subclass
  - Attributes of a superclass are inherited by the subclasses
  - Subclass can have its own specific attributes/relationships
- Every entity in a subclass is a member of its super class(es)



+ ISA Relationship

- Specialization
  - Define a number of subclasses of an entity type
  - Each subclass contains a subset entities of the superclass
  - A subclass can have more specific distinguishing characteristic on entities of the super class
- Generalization
  - Abstraction process of ignoring differences amongst some subclasses and generalize them into a superclass

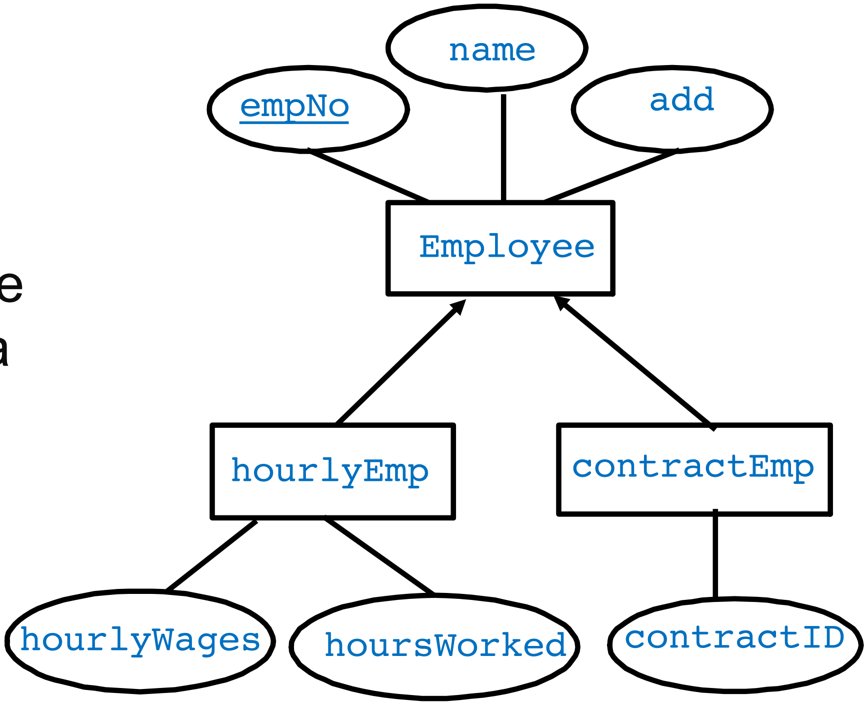


…as in C++ or other OO languages, attributes are inherited

… multiple inheritance possible

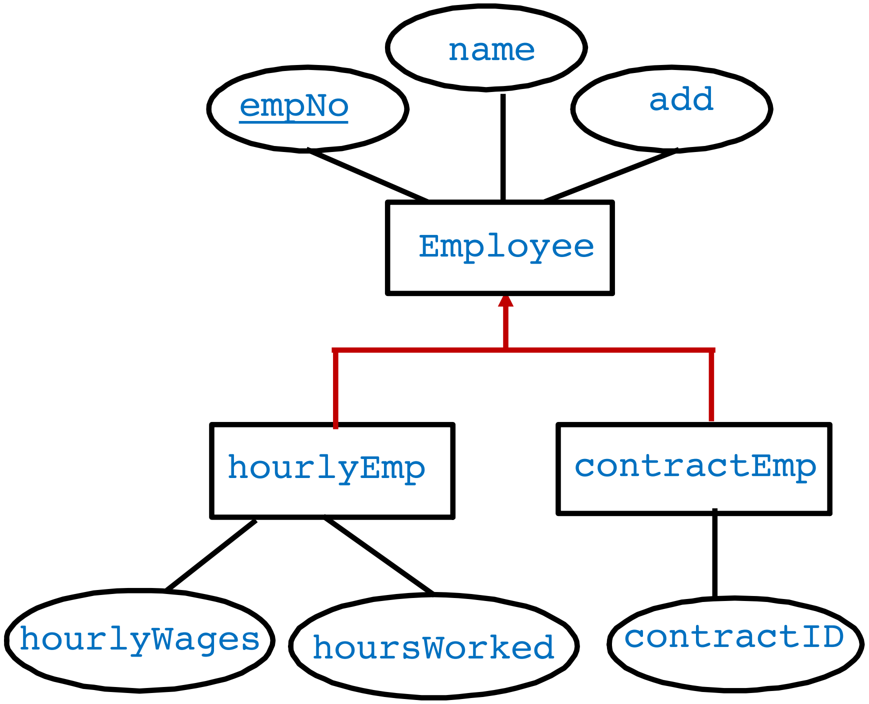
+ Specialization Constraints

- Overlap constraints
  - Can Joe be an Hourly_Emp as well as a Contract_Emp entity?
  - Overlapping or disjoint
- Covering constraints
  - Does every Employee entity have to be either an Hourly_Emp or a Contract_Emp entity?
  - Total or partial participation

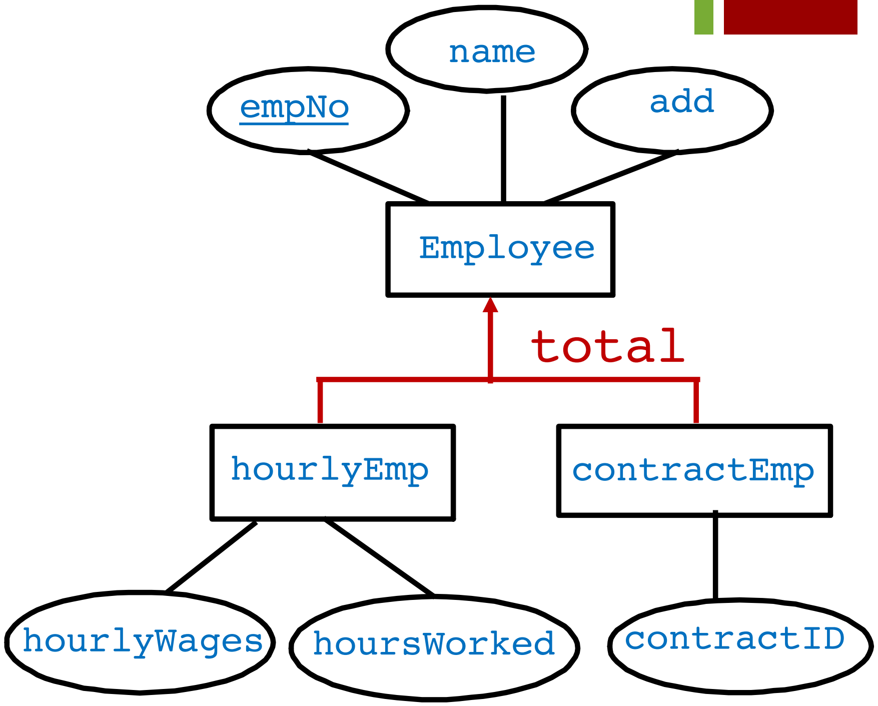


… default option underlined

+ Specialization Constraints (cont)



Disjoint specialization



Total disjoint specialization

+ Exercise

A patient is identified by patient id and Doctors are identified by medical license no. There are two kinds of doctors, GPs and Specialists. For specialists, their area of specialization is also recorded. Doctors work in clinics. One clinic may have several doctors. Clinics are identified by registration number. A patient makes an appointment (date/time) to consult with the doctor. A patient may see the same doctor several times, even on the same date. For each appointment, the doctor’s fee is recorded.

Draw an ER diagram for the above UoD (Universe of Discourse)

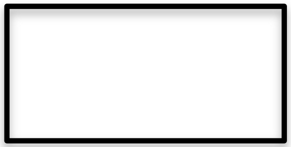
+ Notation Summary

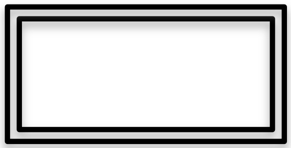
- Naming convention
  - `EntityName` (noun, singular)
  - `RelationshipName` (verb, present tense)
  - `attributeName`
  - `participationRole`
  - Avoid using reserved words or meaningless phrases
- Graphical notations
  - There are several different notation systems, which are equivalent and easy to map between them

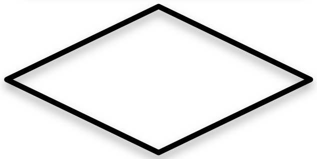
...you can use anything to draw ER diagrams, such as Power Point, Visio, or just hand drawn

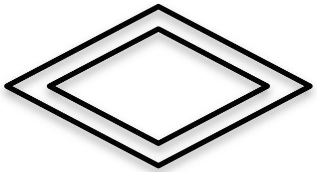
…there are some tools , such as a free tool, draw.io (visit www.draw.io for more information), or Oracle SQL Developer

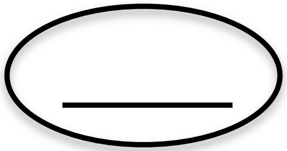
+ Summary of ER Notations (I)

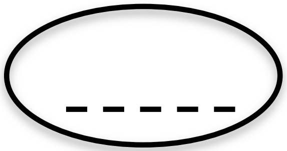

entity

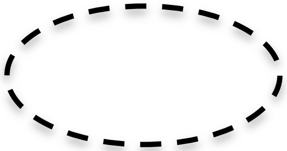

weak entity

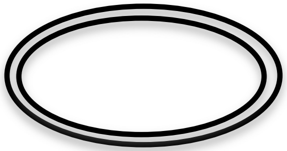

relationship


identifying relationship


attribute (and primary key)

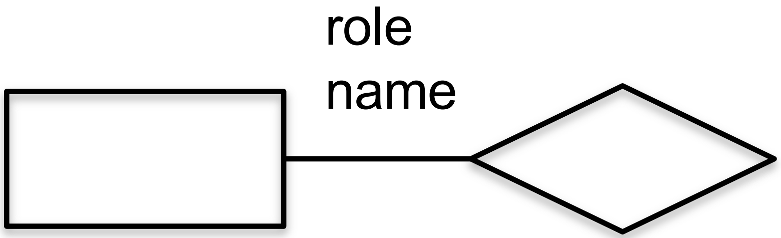

attribute (discriminating attribute of weak entity)


derived attribute

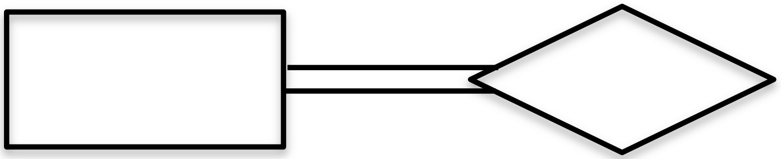

multivalued attribute

… there are several alternative ER notations, but they are equivalent

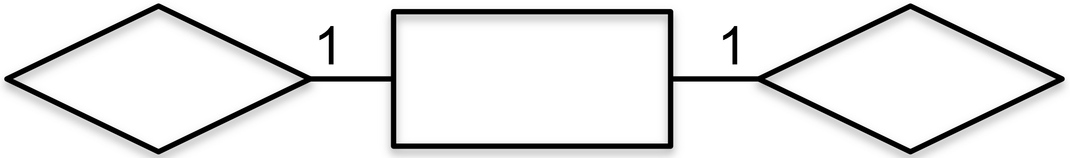
+ Summary ER Notations (II)



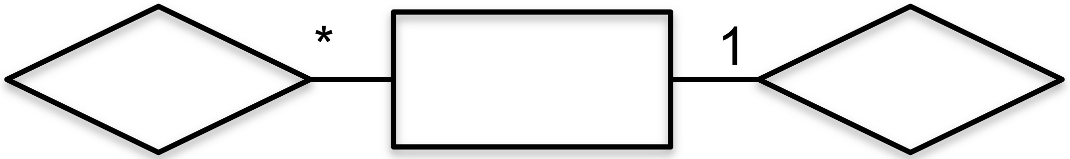
role indicator



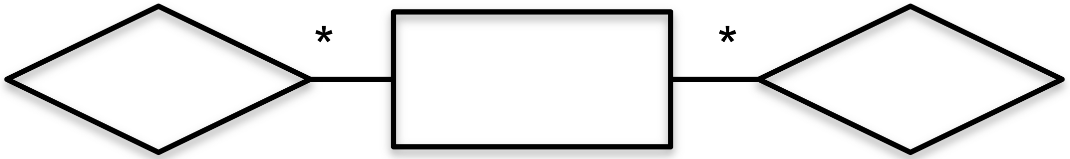
total participation



1-to-1 relationship

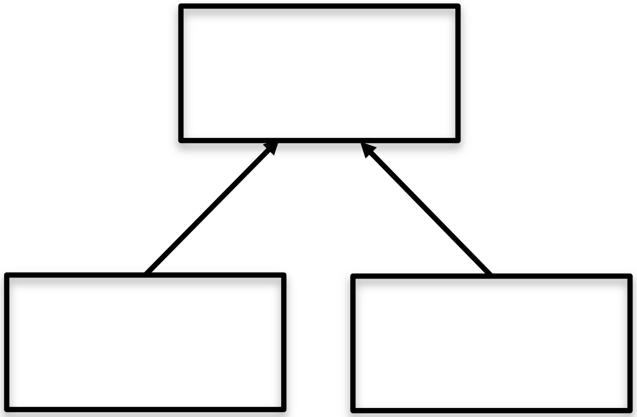


many-to-1 relationship

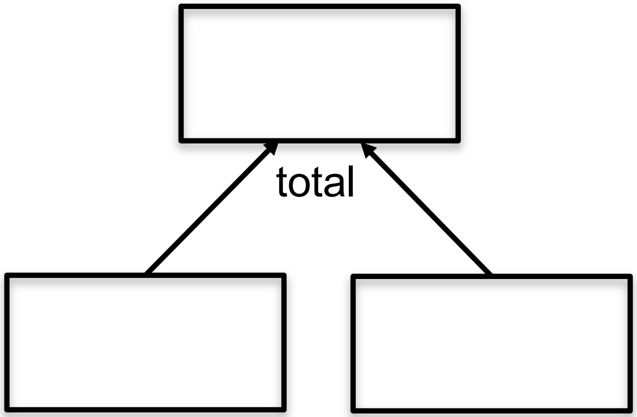


many-to-many relationship

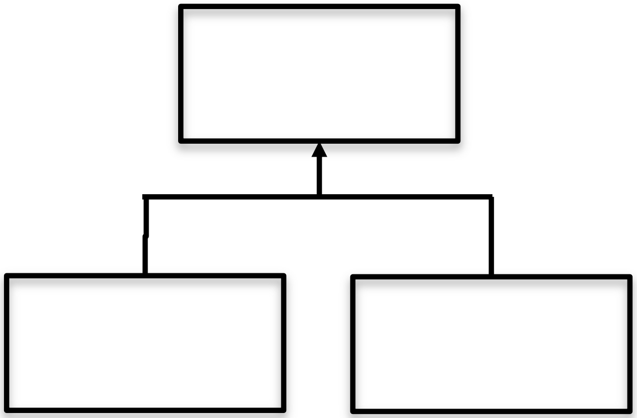
+ Summary of EER Notations



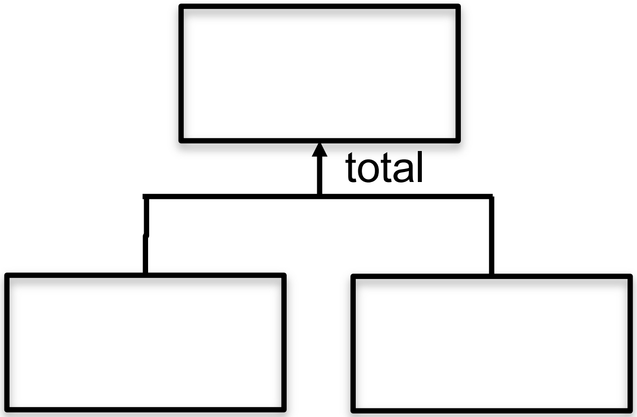
overlapping generalization



total overlapping generalization



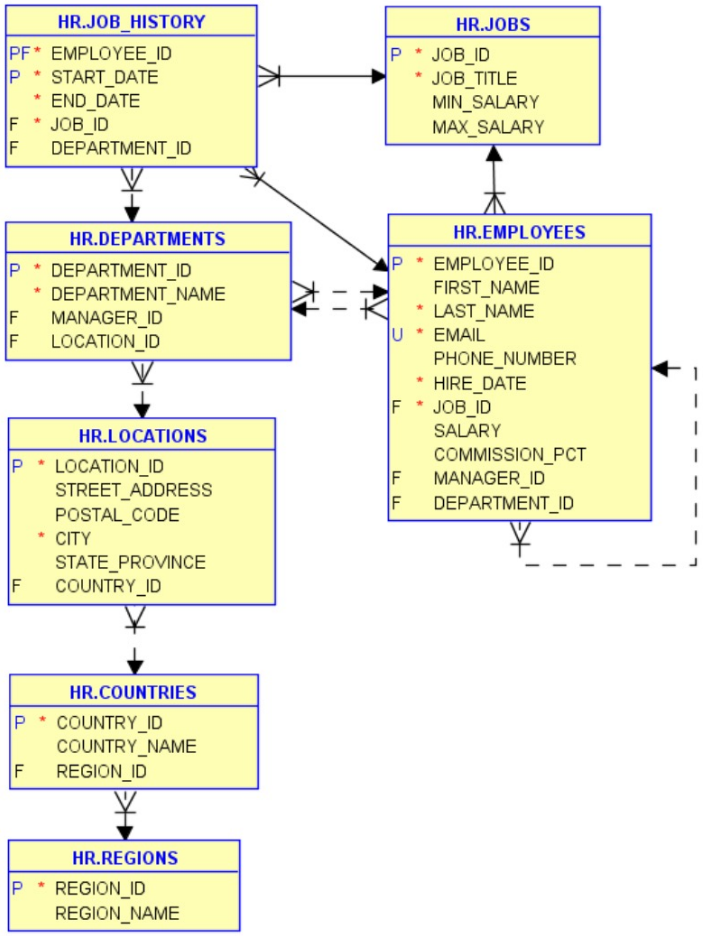
disjoint generalization



total disjoint generalization

+ Alterative Notations

- What we used here are easy for hand drawing
- When generated by a software (e.g., Oracle) →
- Major differences: entities, mappings, participation constraints, and specialization



HR.JOB_HISTORY
PF * EMPLOYEE_ID
P * START_DATE
* END_DATE
F * JOB_ID
F DEPARTMENT_ID

HR.JOBS
P * JOB_ID
* JOB_TITLE
MIN_SALARY
MAX_SALARY

HR.DEPARTMENTS
P * DEPARTMENT_ID
* DEPARTMENT_NAME
F MANAGER_ID
F LOCATION_ID

HR.EMPLOYEES
P * EMPLOYEE_ID
FIRST_NAME
* LAST_NAME
U * EMAIL
PHONE_NUMBER
* HIRE_DATE
F * JOB_ID
SALARY
COMMISSION_PCT
F MANAGER_ID
F DEPARTMENT_ID

HR.LOCATIONS
P * LOCATION_ID
STREET_ADDRESS
POSTAL_CODE
* CITY
STATE_PROVINCE
F COUNTRY_ID

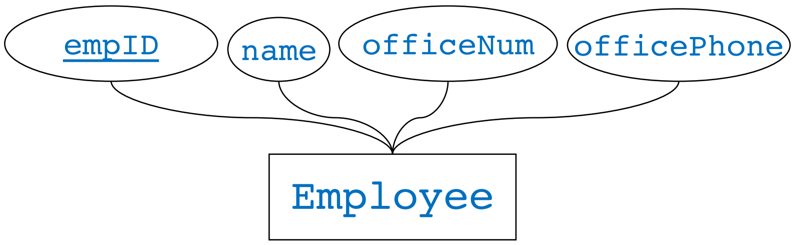
HR.COUNTRIES
P * COUNTRY_ID
COUNTRY_NAME
F REGION_ID

HR.REGIONS
P * REGION_ID
REGION_NAME

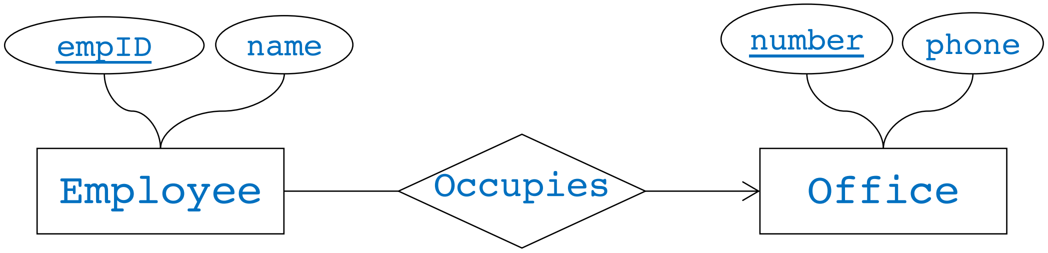
+ Attribute or Entity?

- Choosing between an entity and an 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



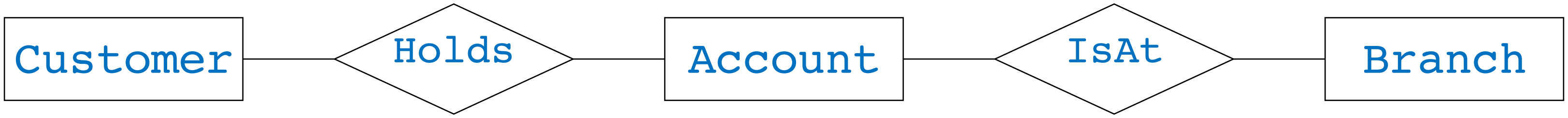
Office as entity



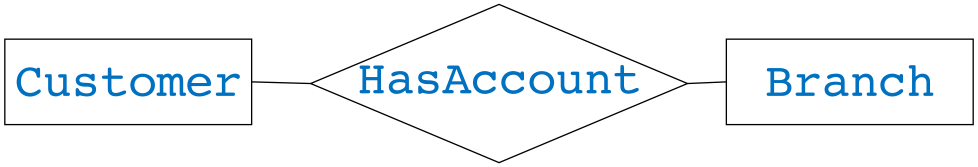
+ Entity or Relationship?

- Choosing between an entity and a 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



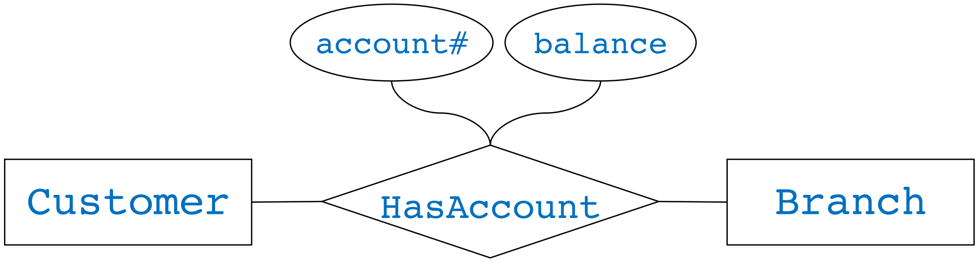
Account as relationship

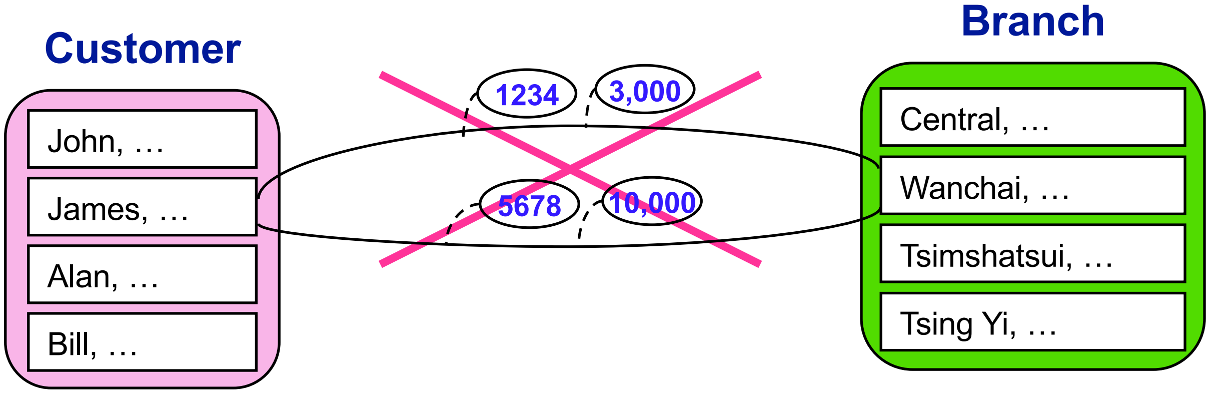


What if you want to have several accounts for a customer?

+ Entity or Relationship?

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

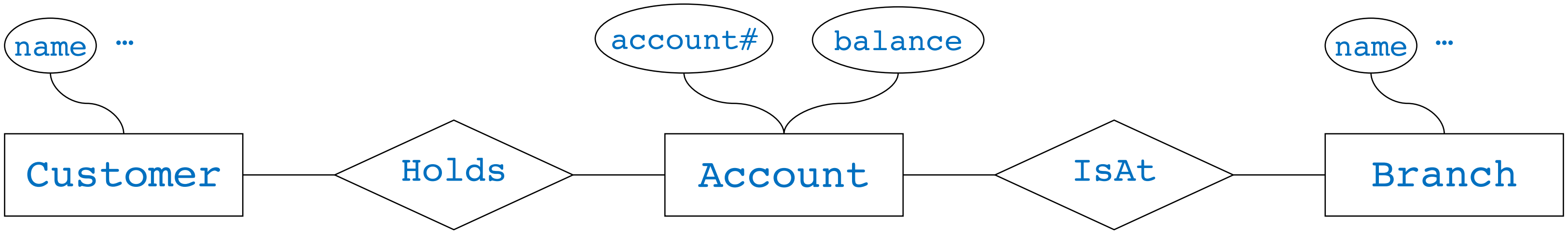


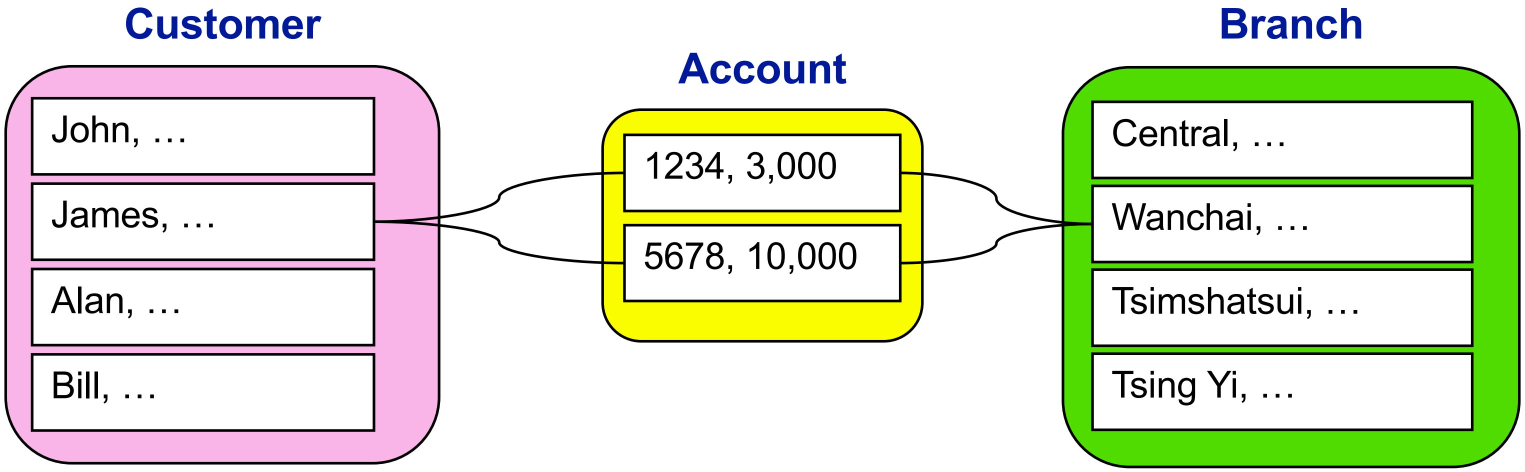


An entity can be related to another entity at most once by a given relationship

+ Entity or Relationship?

- We need to use an entity for Account!

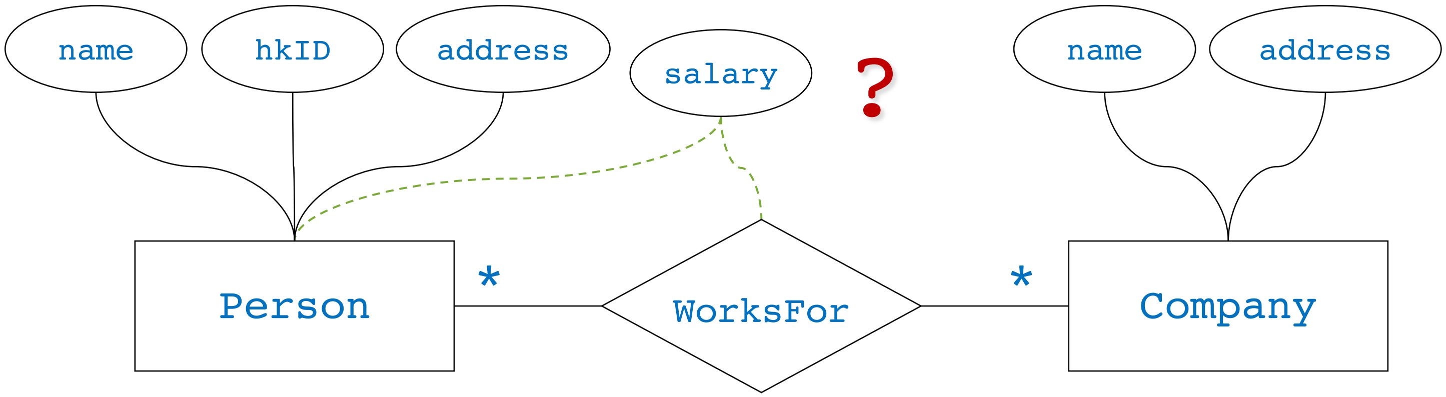




There can be only one relationship instance of a given relationship type between the same two entity instances.

+ Attributes Placement

- Choosing where to place an attribute



Where to place `salary`?

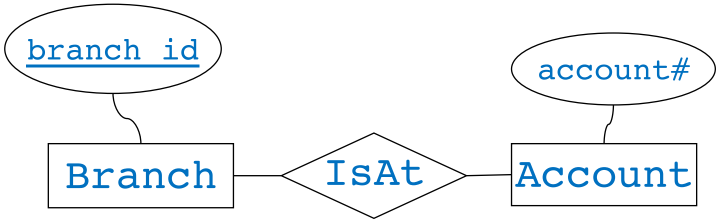
Relationship attributes are usually needed only for many-to-many relationships!
(But can also be used in one to one and one to many relationships)

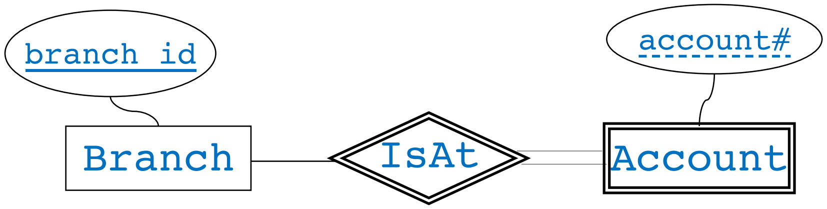
+ Strong or Weak Entity?

- Choosing between a strong entity and a 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?





+ UML

- A software system design includes
  - Data modelling (using the ER model is a common way)
  - User interactions with the system
  - Functional modules and their interactions
- The Unified Modelling Language (UML) is an OMG standard for creating specifications of various components of a software system
  - Class diagram – very similar to the ER diagram
  - Use case diagram – between users and the system
  - Activity diagram – between components for tasks
  - Implementation diagram – between components

…OMG: Object Management Group

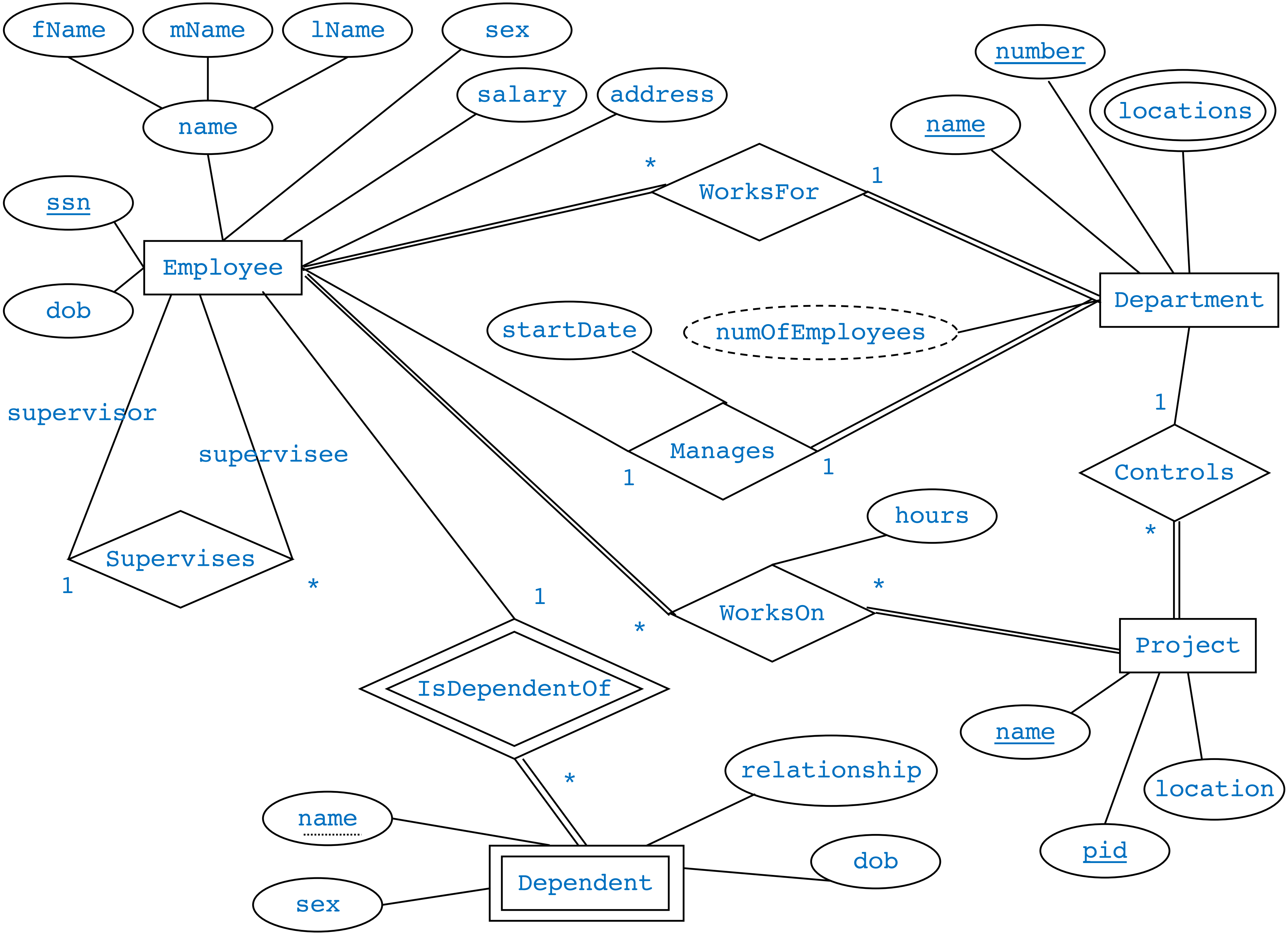
+ Summary

- ER Model is used to support database design before implementation
  - Two components in ERM: Entities and Relationships
- Entity: Type, Set, Attributes (underline the key)
  - Icon: labelled rectangles and ellipses
  - Special cases of attributes: multi-valued, composite, derived
- Relationship: Type, Set, Attributes (underline the key)
  - Icon: labelled diamond and ellipses
  - Cardinality Constraints, Participations, Roles describe a relationship
  - Special cases of relationships: weak, ISA

+ Readings

- Textbook: Chapter 7 (ed6), or Chapter 6 (ed7)

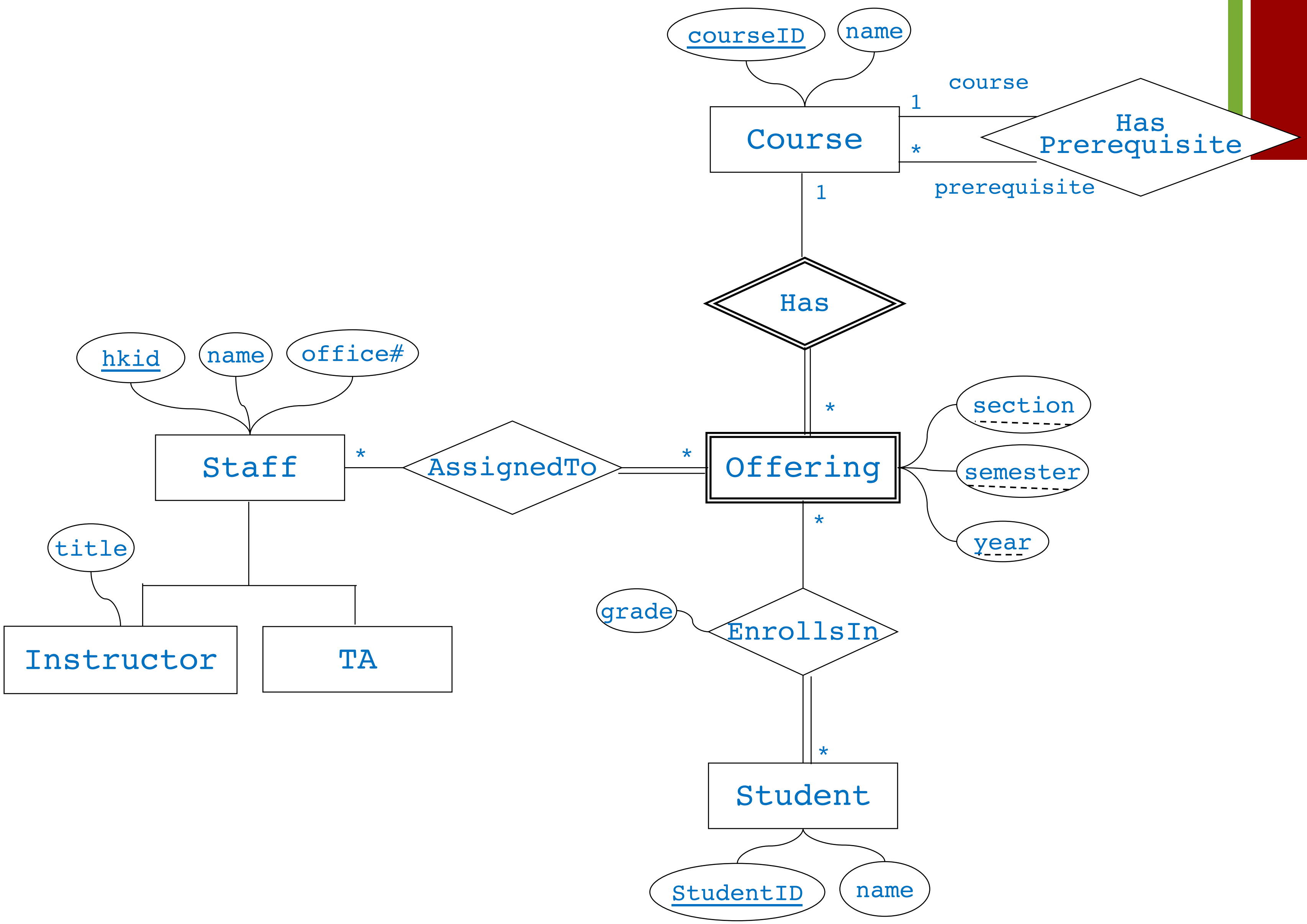
+ Additional Examples


fName
mName
lName
sex
name
salary
address
number
name
locations
ssn
Employee
dob
*
WorksFor
1
Department
startDate
numOfEmployees
supervisor
supervisee
Manages
1
1
1
Controls
*
hours
Supervises
1
*
1
*
WorksOn
*
Project
IsDependentOf
name
*
relationship
location
name
pid
dob
Dependent
sex

+ HKUST COURSE DATABASE

- For each student we store the student id and name
- For each course we store a unique course id, name and none or many prerequisites
- For each offering of a course we store the section, semester and year
- For each staff assigned to a course offering's teaching team we store the hkid, name and office number
- Each student must enroll in one or more course offerings
- Each course offering can enroll none or many 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 (but not both)
- For each instructor we store their academic title (i.e., professor, lecturer, etc.)

+ HKUST COURSE DATABASE: ERD


courseID
name
course
1
Course
Has
Prerequisite
*
prerequisite
1
Has
hkid
name
office#
*
section
semester
year
Staff
*
AssignedTo
*
Offering
*
title
grade
EnrollsIn
Instructor
TA
*
Student
StudentID
name