Week 4 – The Entity Relationship Model

Australian National University

W4 - Housekeeping

Assignment 1 on SQL **is available now** on Wattle (Thanks Patrik)
Due at 23:55 28th Mar. You will have 2 weeks' time to finish.

- No extensions will be given.
- Get a start on it early and submit early—be kind to yourself.
- **Must** be done individually. No group work allowed.
- Do not post solutions, results, interpretations on the Wattle forum.
- To be fully correct, your SQL queries should produce correct answers when tested against other databases that have the same schema, but different content.
- Partial marks may be given if a query only has minor issues, but any query that is not runnable (i.e., that has syntax errors or is not correct w.r.t. the database schema) will receive zero marks.
- For reference: sample questions with solutions

Week 4

- Quiz Feedback
- Database Design (4 phases)
- Basic Concepts in ER Model
- Conceptual Design (2nd phase)
- Logical Design (3rd phase)

Australian National University

Quiz-Q5

Q5 Version 1 (latest)

Question 1

Not yet answered

Marked out of 0.10

When converting a binary M:N relationship from an ER model into a relation schema, the M:N relationship will become a separate table with the primary key as the combination of all the attributes of the participating entity types.

Select one:

- ○ True
- ○ False

Australian National University

Quiz-Q6 statistics

Analysis of responses

Part of question	Response	Partial credit	Count	Frequency
102655918	Entity-Relationship Model (Part 1): Database Design Process	0.00%	37	9.71%
102655919	Entity-Relationship Model (Part 2): Basic Modeling Concepts	0.00%	59	15.49%
102655920	Entity-Relationship Model (Part 3): Enhanced Modeling Concepts	0.00%	132	34.65%
102655921	Entity-Relationship Model (Part 4): From ER to Relations	0.00%	178	46.72%
102655922	None of the above	100.00%	126	33.07%

Week 4

- Quiz Feedback
- Database Design (4 phases)
- Basic Concepts in ER Model
- Conceptual Design (2nd phase)
- Logical Design (3rd phase)

Database Design – Four Phases

- The database design process has **four phases**:

1. **Requirements Collection and Analysis**
2. **Conceptual Design**
   Entity-Relationship Model
3. **Logical Design**
   From Entity-Relationship Model to Relation Schemas
4. **Physical Design**

Week 4

- Quiz Feedback
- Database Design (4 phases)
- Basic Concepts in ER Model
- Conceptual Design (2nd phase)
- Logical Design (3rd phase)

Entity-Relationship (ER) Model

- A model is a simplification of reality, often represented as a graphical depiction of data. It helps us understand, specify, and build a system.
- The components of ER Model:
  - **Entity:** "Things" in the real world (with independent existence).

    e.g., Student, Course
  - **Relationship** Associations between entities.

    e.g., a student enrolls in one course
  - **Attributes:** Properties that describe entities and relationships.

    e.g., Student has attributes like StudentID, Name, Age
  - **Key attributes:** Attribute(s) that identifies an entity instance uniquely.
    e.g., StudentID uniquely identifies a Student

Keys

- The definitions for **superkey/primary key/candidate key** of an entity type is the same as for a relation schema.
  - A **superkey** of an entity type is a set of one or more attributes whose values uniquely determine each entity in an entity set.
  - A **candidate key** of an entity type is a minimal (in terms of number of attributes) superkey.
  - For an entity type, several candidate keys may exist. During conceptual design, one of the candidate keys is selected to be the **primary key** of the entity type.
- A **primary key** of a relationship type is the combination of primary keys of the entity types that participate in the relationship type.

Constraints on Relationships

- Below are useful constraints in describing binary relationship types:
  - **Cardinality ratios**
    - Specifies the *maximum* number of relationships that an entity can participate in.
  - **Participation constraints** (total, partial)
    - Specifies whether the existence of any entity depends on its being related to another entity via the relationship type.

Constraints on Relationships - Cardinality Ratios

- **Many-To-Many**

Meaning: An employee can work for many departments (≥ 0), and a department can have several employees.

- **One-To-Many**

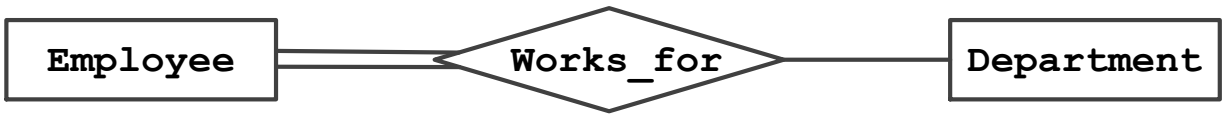
Meaning: An employee can work for at most one department (≤ 1), and a department can have several employees.

- **One-To-One**

Meaning: An employee can work for at most one department, and a department can have at most one employee.

Constraints on Relationships - Participation constraints

- **Total**



Meaning: Each employee must work for a department and each department may or may not have employees.

- **partial** (default)

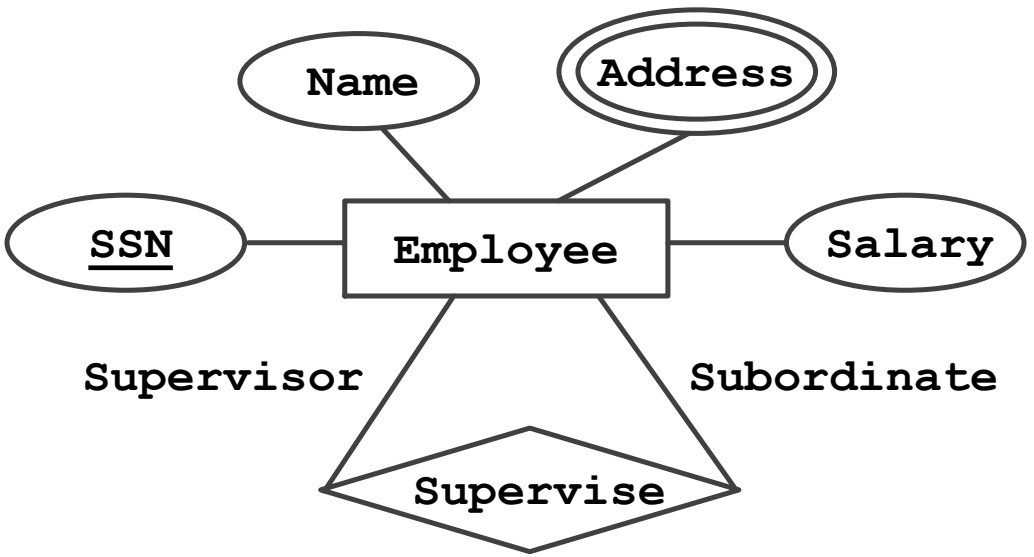
Meaning: An employee may or may not work for a department and each department may or may not have employees.

Recursive Relationships

- **Recursive relationships**

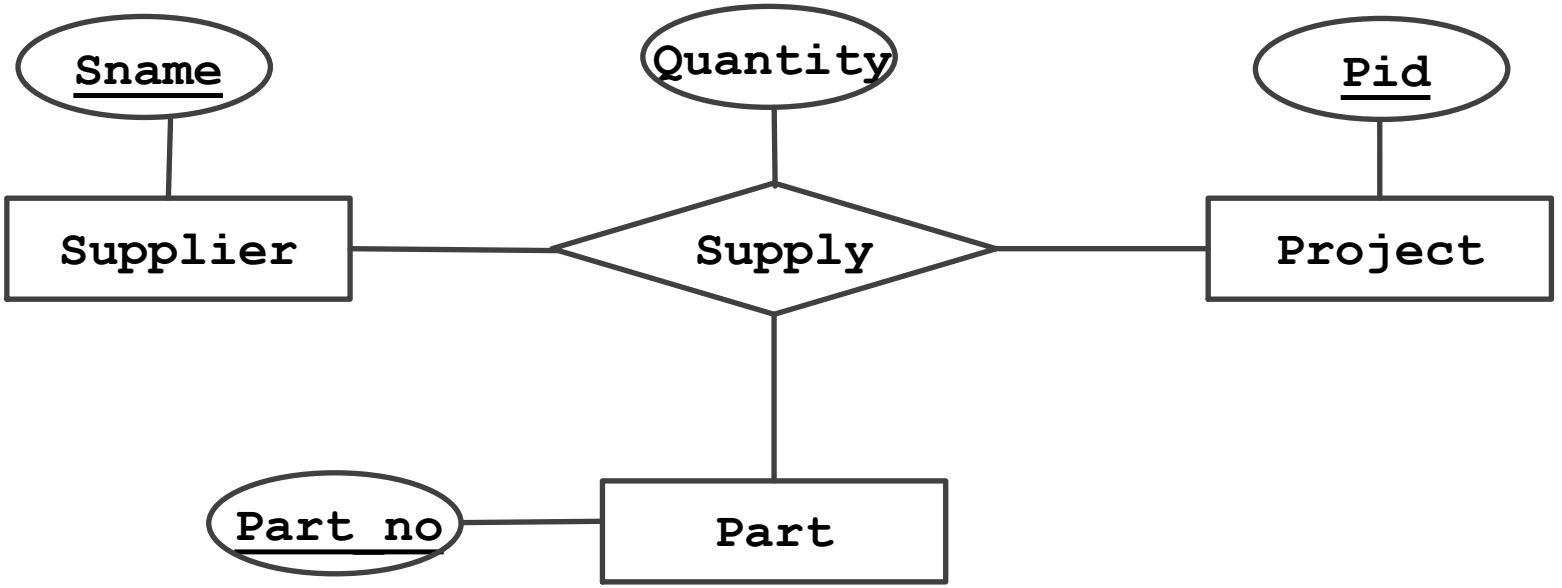
  Same entity type can participate more than once in a relationship type in different roles, e.g., marriage between persons and parent-child between persons

- A **role name** signifies the role that a participating entity plays in each relationship.



Higher-Degree Relationship Types

- We may use higher-degree relationship types to model more complicated relationships, i.e., involving multiple entity types.



Enhanced Entity-Relationship (EER) Model

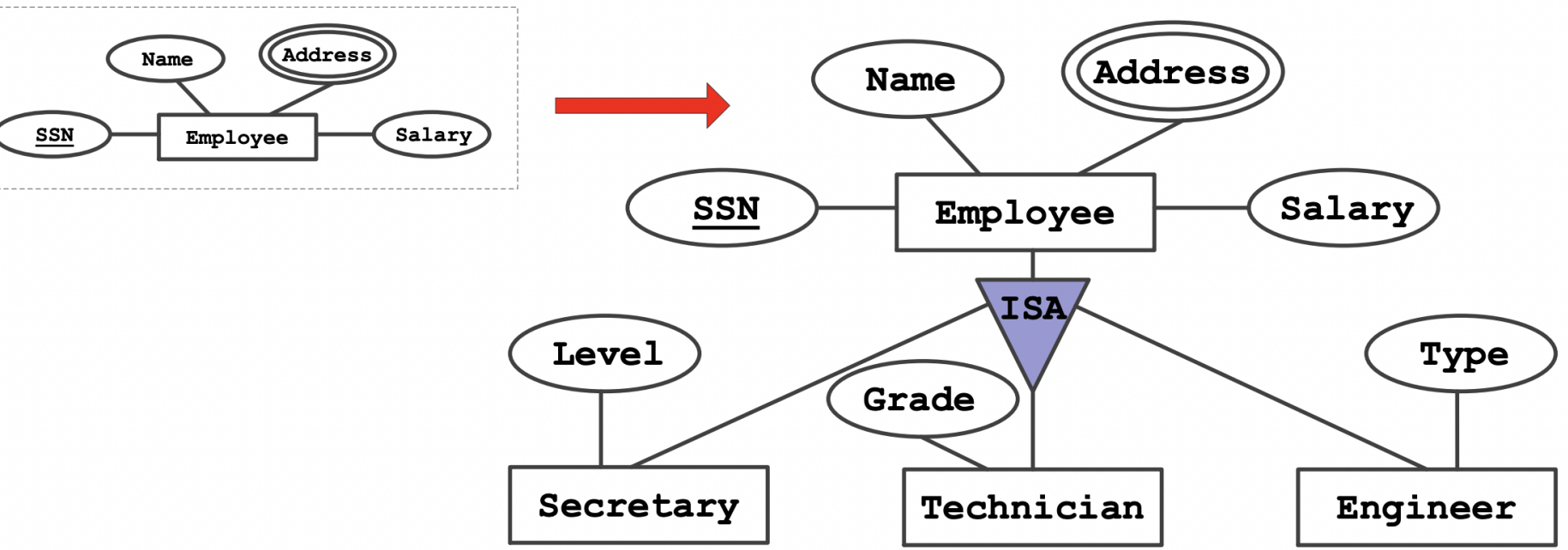
- The basic modelling concepts are only sufficient for some database applications.
- To reflect data properties and constraints more precisely, a number of enhanced ER models (EERs) were proposed.
- Each EER model includes all the basic modeling concepts of the ER model we discussed before.
- We will further discuss the following concepts in EERs:
  - **Subclass/superclass**
  - **Specialisation/generalisation**
  - **Constraints on specialisation/generalisation**

Subclass and Superclass

- **Subclass of an entity type:** subgrouping of entities.
  - In many cases subclasses need to be **represented explicitly** because of their application significance.
- Superclass/subclass, Supertype/subtype and Class/subclass are different names for the same concept.
  - Subclass inherits attributes and relationships of superclass.
  - Subclass can have additional attributes and relationships.
- This type of relationship between subclass and superclass is often described as an **ISA relationship type**.

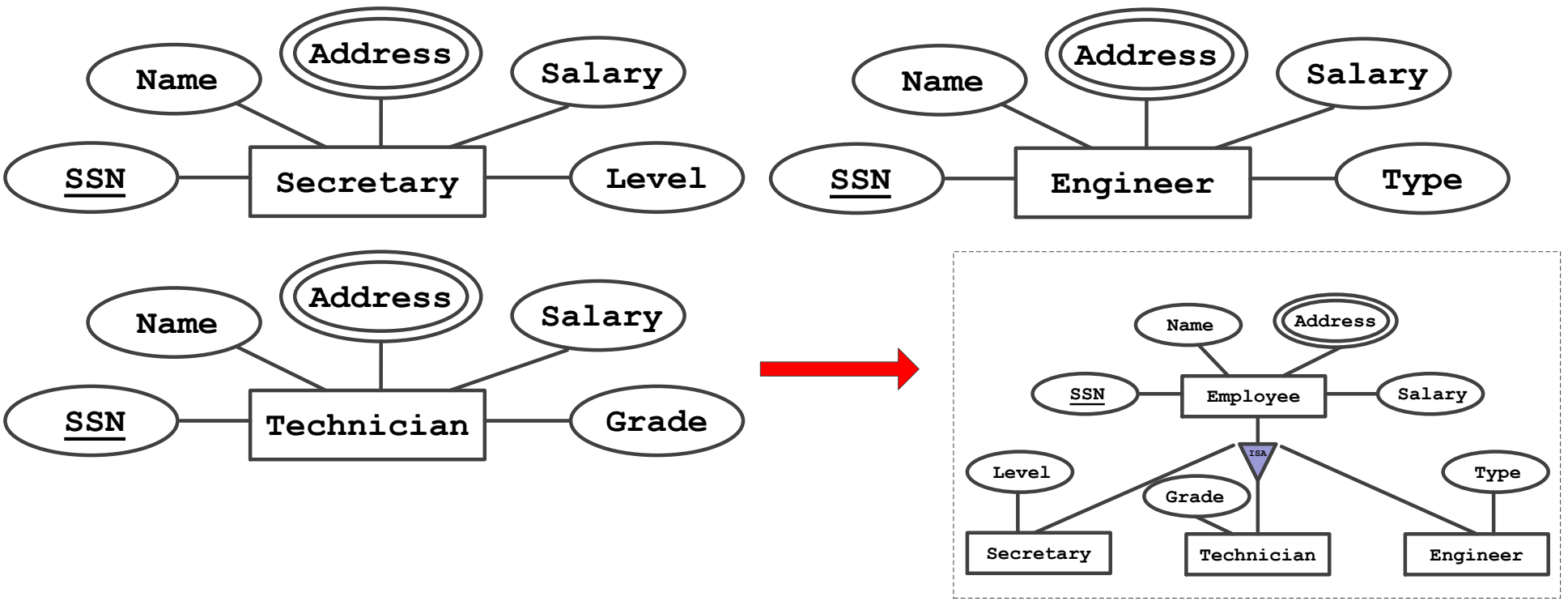
Specialisation and Generalisation

- **Specialization** is the process of defining a set of subclasses of an entity type (top-down).
  - Defined on distinguishing features of entities in the superclass, e.g., based on the *job type* of each employee:



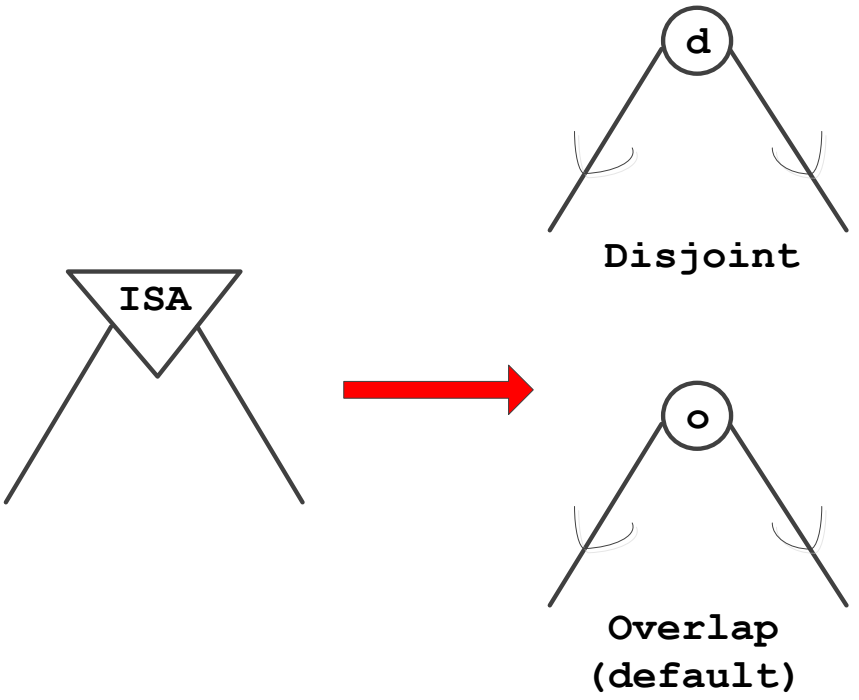
Specialisation and Generalisation

- **Generalization** is a reverse process of specialization (bottom-up).
  - Common features of entities in subclasses may be generalized into single superclass (including primary key).



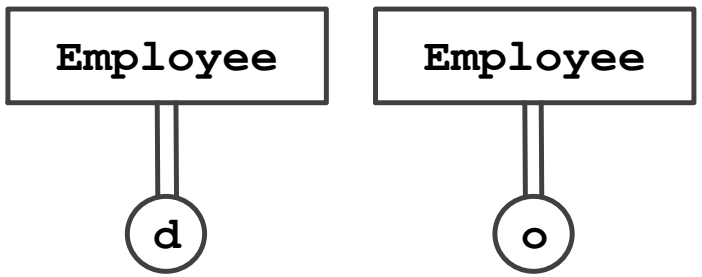
Constraints on Specialisation and Generalisation

- **Disjointness constraint**
  - Specifies that the subclasses of the specialization must be **disjoint**.
  - If not constrained, then entities in the subclasses may **overlap**.

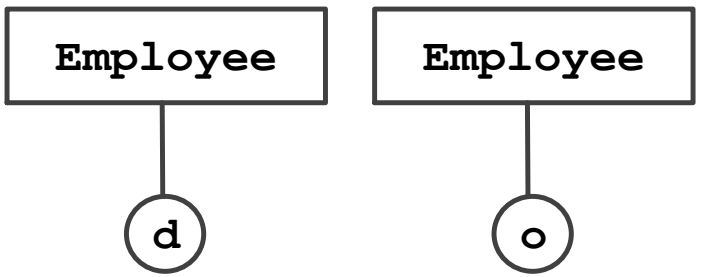


Constraints on Specialisation and Generalisation

- **Completeness constraint**
  - **total** – *every* entity in the superclass must be a member of at least one subclass.
  - **partial** – an entity may not belong to any of the subclasses.



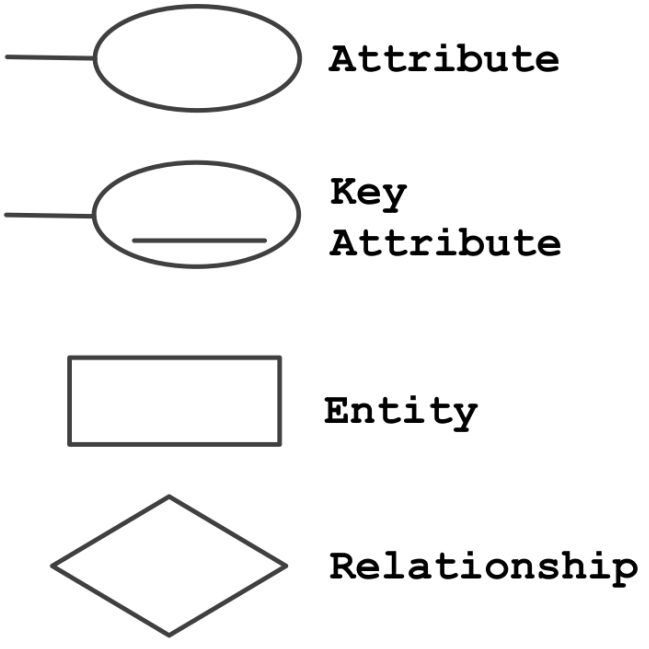
Total



Partial
(default)

Entity-Relationship (ER) Diagram

- ER diagram: diagrammatic notation associated with the ER model.
- ER diagrams (Peter Chen in 1976):
  - **Attribute** as *oval*;
  - **Key attribute** with *underlined*;
  - **Entity** as *rectangles*;
  - **Relationship** as *diamonds*.



Software tool to draw ER diagram

- We require students to use an academic tool, TerraER, to draw the ER diagrams.
- TerraER allows you to save your ER diagrams into xml files and export your ER diagrams as a JPEG figure.
- The jar file with the instruction on how to use it will be on your lab 4 material.

Week 4

- Quiz Feedback
- Database Design (4 phases)
- Basic Concepts in ER Model
- Conceptual Design (2nd phase)
- Logical Design (3rd phase)

How to construct an ER or EER Model?

- Draw an ER or EER diagram to represent the following design:
  1. Identify the entity types (including weak entity types)
  2. Identify the relationship types
  3. Identify the attributes of entity and relationship types
  4. Identify a primary key for each entity type
  5. Identify cardinality ratios and participation constraints on relationships
  6. Determine the disjointness and completeness constraints for each ISA

Exercise 1

- The university keeps track of each student's name, student number, social security number, address, phone, and birthdate. Both social security number and student number have unique values for each student.
- Each student has exactly one major, and may have a minor (if any) with departments.
- Each department has name, department code, office number, office phone, and college. Both name and code have unique values for each department.
- Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.
- Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
- A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

Each student has name, student number, social security number, address, phone and birthdate. Both social security number and student number have unique values for each student.

- What are the entities, relationships and attributes?

  Entities: STUDENT

  Relationships:

  Attributes: name, student number, social security number, address, phone and birthdate for STUDENT

Each student has exactly one major, and may have a minor (if any) with departments

- What are the entities, relationships and attributes?

  Entities: STUDENT, DEPARTMENT

  Relationships: has_major_with between STUDENT and DEPARTMENT, has_minor_with between STUDENT and DEPARTMENT

  Attributes: name for has_major_with, name for has_minor_with

Each student has exactly one major, and may have a minor (if any) with departments.

- What are the constraints on relationship "**has_major_with**"?
  - **Cardinality ratios**:
    Every student has at most **one** major and a department may offer **many** majors (to different students);
  - **Participation constraints**:
    Every student **must** have one major (**total**) and each department **must** (typically) offer one major (**total**);
- What are the constraints on relationship "**has_minor_with**"?
  - **Cardinality ratios**:
    Every student has at most **one** minor and a department may offer **many** minor (to different students)
  - **Participation constraints**:
    Every student **may or may not** have one minor (**partial**) and each department **must** (typically) offer one minor (**total**).

Each department has name, department code, office number, office phone, and college. Both name and code have unique values for each department.

- What are the entities, relationships and attributes?

  Entities: DEPARTMENT

  Relationships:

  Attributes: name, department code, office number, office phone, and college for DEPARTMENT

Each course has a course name, description, course number, number of semester hours, level, and offering department.

- What are the entities, relationships and attributes?
  - **Entities:** course, department
  - **Relationships:** offer (between **department** and **course**)
  - **Attributes**: course name, description, course number, number of semester hours and level (of the entity **course**)
- What are the constraints on relationship "**offer**"?
  - **Cardinality ratios**: Every course is offered by at most **one** department and a department may offer **many** courses
  - **Participation constraints**: Every course **must** be offered by some department (**total**) and each department **may (or may not)** offer any courses (**partial**).

Each section of a course has an instructor, semester, year, and section number and the section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.
A grade record refers to each student and a particular section, consisting of a final mark and a letter grade from (F, D, C, B, A).

- What are the entities, relationships and attributes?

  Entities: section, course, student

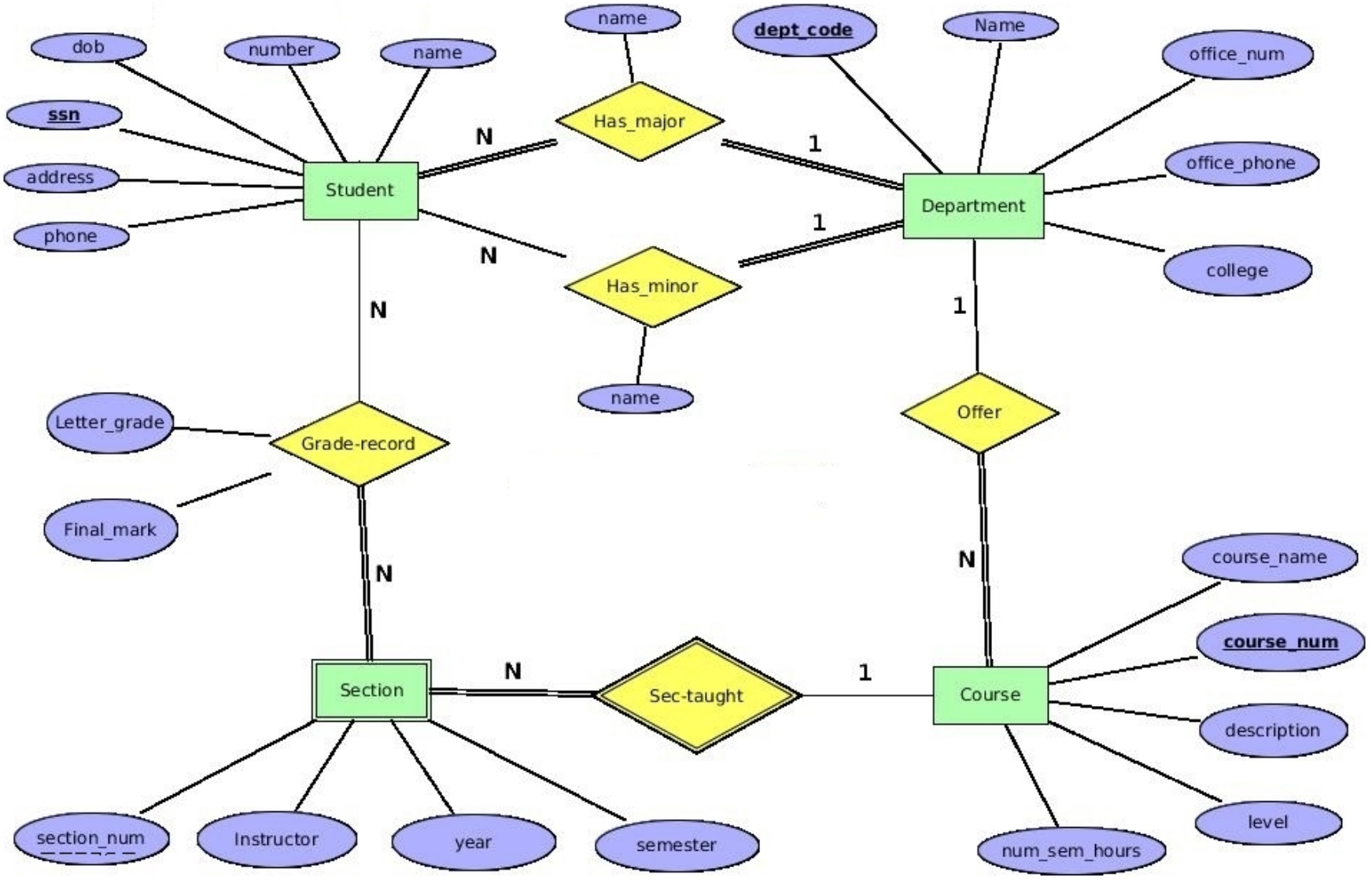
  Relationships: section_taught (between **section** and **course**), grade_record (between **student** and **section**)

  Attributes: instructor, semester, year, and section number (of the **weak** entity **section**), final mark and letter grade (of the relationship **grade_record**)

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
  student, course, department, section (weak entity)

- Identify the relationships
  - **has_major** (between **student** and **department**)
  - **has_minor** (between **student** and **department**)
  - **offer** (between **department** and **course**)
  - **section_taught** (between **section** and **course**)
  - **grade_record** (between **student** and **section**)
- Identify the attributes of entities and relationships and identify a primary key for each entity type
- Identify cardinality ratios and participation constraints on relationships


name
dept_code
Name
dob
number
name
office_num
ssn
Has_major
N
1
address
Student
Department
office_phone
phone
1
N
Has_minor
college
N
1
name
Letter_grade
Offer
Grade-record
Final_mark
N
N
course_name
course_num
Section
N
Sec-taught
1
Course
description
level
section_num
Instructor
year
semester
num_sem_hours

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells products in both local shops and webstores on the Internet. Each local shop has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each webstore. Every product has a unique productID, a description, an item price, and a quantity in stock. The database application should also record customers' details such as their name, address and email. Every customer is assigned a unique ID. A customer may place an order that consists of at least one product and each order is from either a shop or a webstore. Customers have three payment options (i.e., cash, paypal, and credit card) but for each order only one payment option can be chosen. A delivery may be requested for each order. After full-payment is received, a delivery would be sent out subject to products' availability. Every delivery has a unique tracking number.

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells **products** in both local **shops** and **webstores** on the Internet. Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**. Every **product** has a unique productID, a description, an item price, and a quantity in stock. The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID. A **customer** may place an **order** that consists of at least one **product** and each **order** is from either a **shop** or a **webstore**. **Customers** have three payment options (i.e., cash, paypal, and credit card) but for each **order** only one payment option can be chosen. A **delivery** may be requested for each **order**. After full-payment is received, a **delivery** would be sent out subject to **products**' availability. Every **delivery** has a unique tracking number.

- Identify the entities (including weak entity types)
  shop, **webstore**, **product**, **customer**, **order**, **delivery**
- Identify subclass/superclass and the corresponding disjointness and completeness constraints
  - The company sells products in both local shops and webstores on the Internet.
  - Each order is associated with either a shop or a webstore.
  - subclass **shop**, **webstore**
  - superclass **store**

(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells **products** in both local **shops** and **webstores** on the Internet. Each local **shop** has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each **webstore**. Every **product** has a unique productID, a description, an item price, and a quantity in stock. The database application should also record **customers**' details such as their name, address and email. Every **customer** is assigned a unique ID. A **customer** may **place** an **order** that **consists of** at least one **product** and each **order** **is from** either a **shop** or a **webstore**. **Customers** have three payment options (i.e., cash, paypal, and credit card) but for each **order** only one payment option can be chosen. A **delivery** may be requested **for** each **order**. After full-payment is received, a **delivery** would be sent out subject to **products**' availability. Every **delivery** has a unique tracking number.

- Identify the relationships
  - **customer** **place** **order**
  - **order** **consists of** **product**
  - each order **is from** **store**(superclass) (either subclass **shop** or subclass **webstore**)
  - **delivery** **is for** **order**

- Identify the attributes of entities and relationships and identify a primary key for each entity type
  - Every product has a unique productID, a description, an item price, and a quantity in stock.
    Attributes for **product**: **productID, description, item price, quantity**
    Primary key for **product**: **productID**
  - The database application should also record customers' details such as their name, address and email. Every customer is assigned a unique ID.
    Attributes for **customer**: **name, address, email, CustomerID**
    Primary key for **customer**: **CustomerID**
  - Each local shop has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each webstore.
    Attributes for superclass **store**: **name**, **location/URL**
    Primary key for superclass **store**: **location/URL**
    Attributes for subclass **shop**: **phone number, email**
    Attributes for subclass **webstore**: **last updated date**

- Identify cardinality ratios and participation constraints on relationships
  - Relationship: A customer may place an order
    Cardinality ratios: A customer may **place** **many** orders and an order **is placed by** **one** customer.
    Participation constraints: A **customer** **may or may not** **place** any orders (**Partial**). An **order** **must** **be placed by** one customer (**Total**).
  - Relationship: Each order consists of at least one product
    Cardinality ratios: An order may **contain** **many** products and an product may **be contained** in **many** orders.
    Participation constraints: An **order** **must** **contain** some product (**Total**). A **product** **may or may not** **be contained** in an order (**Partial**).

- Identify cardinality ratios and participation constraints on relationships
  - Relationship: A delivery may be requested for each order
    Cardinality ratios: A delivery **is for** at most **one** order and an order **has** at most **one** delivery.
    Participation constraints: A **delivery** **must** **be for** an order (**Total**).
    An **order** **may or may not** **have** a delivery (**Partial**).
  - Relationship: Each order is from a store (either a shop or a webstore)
    Cardinality ratios: An order **is from** at most **one** store and a store may **have** **many** orders.
    Participation constraints: An order **must** **be from** a store. A store **may or may not** **have** an order.

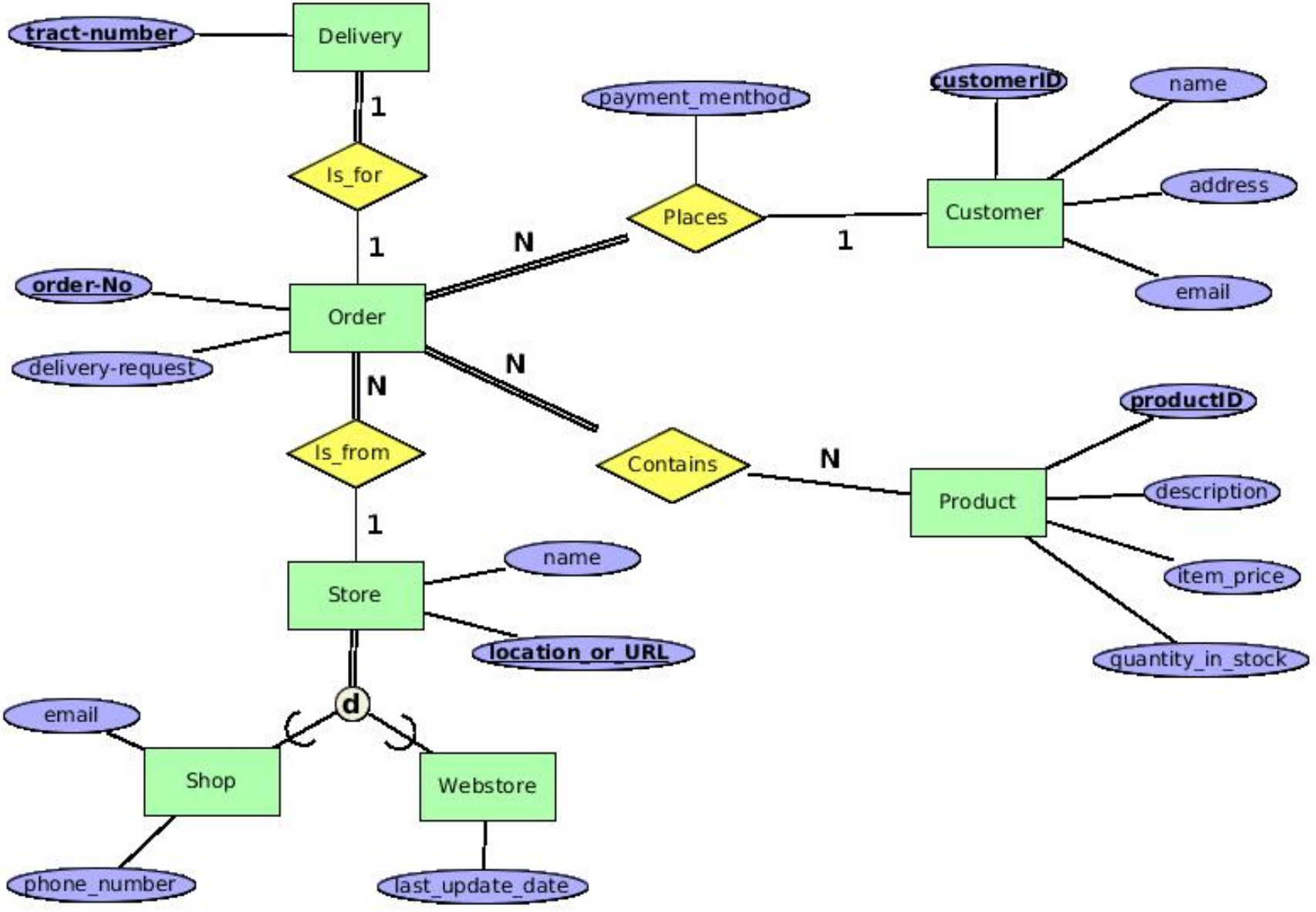
Constructing an ER or EER Model

Constructing an ER or EER Model

- Identify the entities (including weak entity types)
- Identify subclass/superclass
- Identify the relationships
- Identify the attributes of entities and relationships
- Identify cardinality ratios and participation constraints on relationships

- **Not all the constraints can be expressed in the ER model**

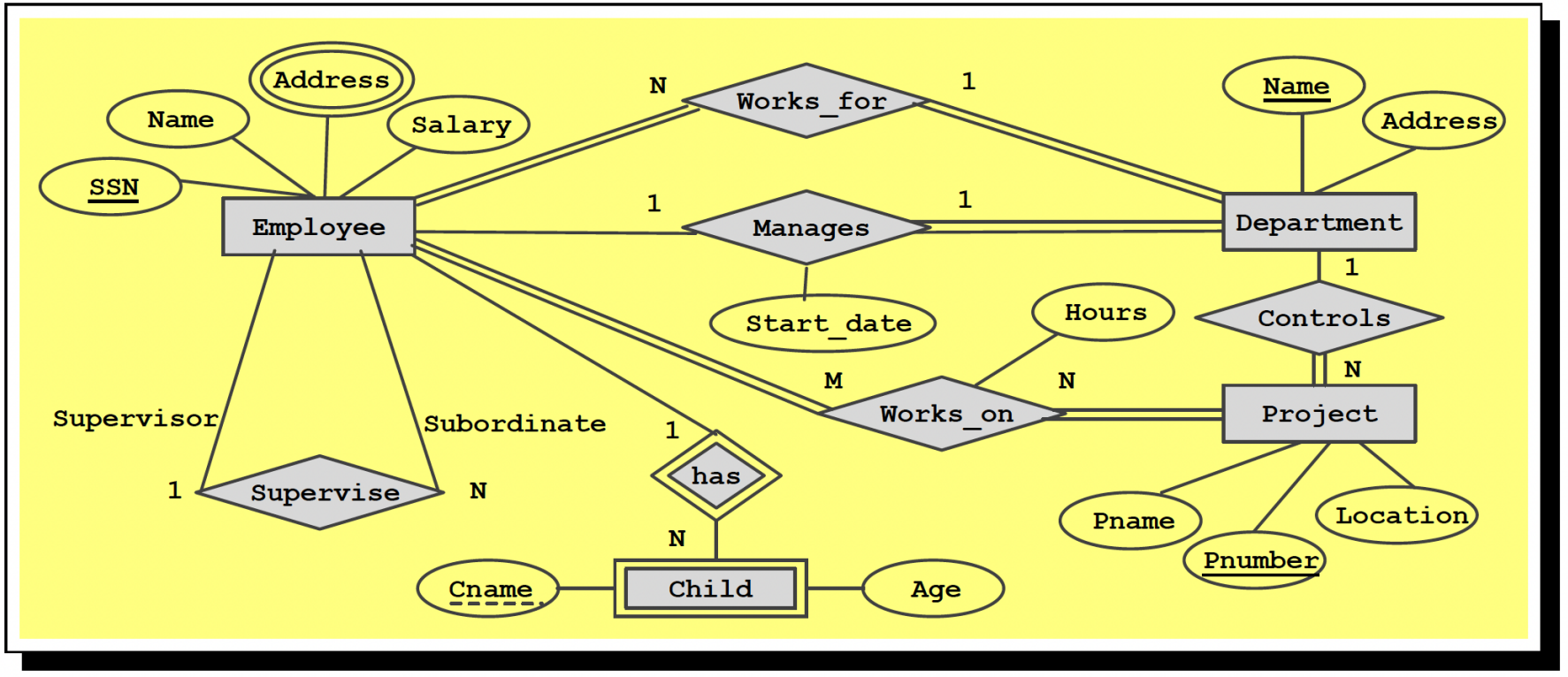
(Exercise 2) A retailer company wants to build a database application for managing information about its sale process. The company sells products in both local shops and webstores on the Internet. Each local shop has a name, contact details (e.g., phone number and email), and a unique location. The database application also needs to store the URL(unique), name and last updated date of each webstore. Every product has a unique productID, a description, an item price, and a quantity in stock. The database application should also record customers' details such as their name, address and email. Every customer is assigned a unique ID. A customer may place an order that consists of at least one product and each order is from either a shop or a webstore. **Customers have three payment options (i.e., cash, paypal, and credit card) but for each order only one payment option can be chosen.** A delivery may be requested for each order. **After full-payment is received, a delivery would be sent out subject to products' availability**. Every delivery has a tracking number.


tract-number
Delivery
1
Is_for
1
N
payment_menthod
Places
1
customerID
name
address
email
Customer
order-No
delivery-request
Order
N
N
Is_from
1
Contains
N
productID
description
item_price
quantity_in_stock
Product
name
Store
location or URL
d
email
Shop
Webstore
phone_number
last_update_date

Week 4

- Quiz Feedback
- Database Design (4 phases)
- Basic Concepts in ER Model
- Conceptual Design (2nd phase)
- Logical Design (3rd phase)

An ER Diagram - The Company Database



ER-to-Relations Algorithm

- 7-step algorithm to convert the basic ER model into relations, and more steps for the EER model.

  Step 1: Mapping of Regular Entity Types

  Step 2: Mapping of Weak Entity Types

  Step 3: Mapping of Binary 1:1 Relationship Types

  - Foreign key approach
  - Merged relation approach
  - Cross-reference approach

  Step 4: Mapping of Binary 1:N Relationship Types

  Step 5: Mapping of Binary M:N Relationship Types

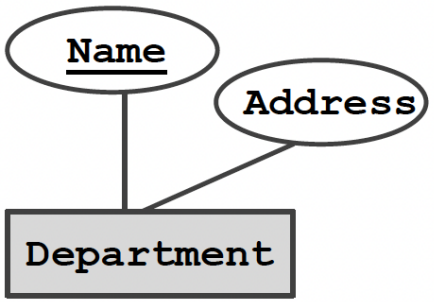
  Step 6: Mapping of Multi-valued Attributes

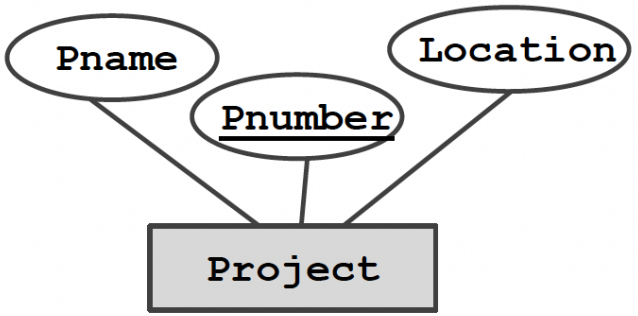
  Step 7: Mapping of N-ary Relationship Types

  Step 8: Mapping of Superclass/Subclass

Step 1: Regular Entity types

- For each regular entity type E, **create a relation schema** with the attributes of E (ignore multi-valued attributes until Step 6), where
  - **PK:** the key attributes of E

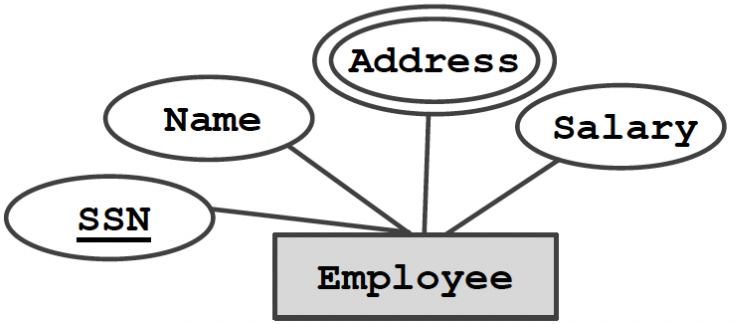




- DEPARTMENT(Name, Address) with PK: {Name}
  PROJECT(Pnumber, Pname, Location) with PK: {Pnumber}
- **Note:** These are not necessarily the final relation schemas of DEPARTMENT and PROJECT.

Step 1: Regular Entity types

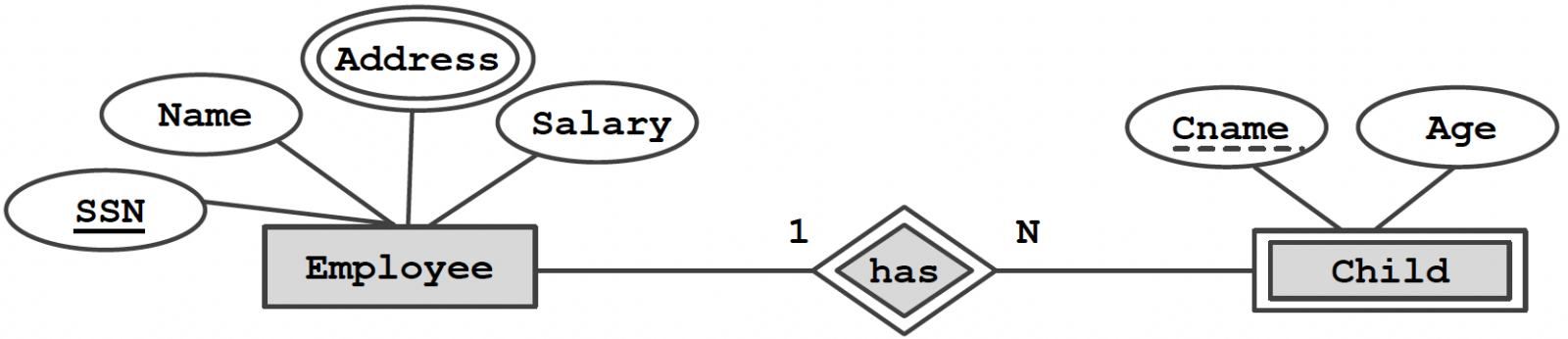
- How can we translate the regular entity type EMPLOYEE?



- EMPLOYEE(SSN, Name, Salary) with PK: {SSN}
- **Note:**
  - This is not the final relation schema of EMPLOYEE (will be further extended later on).
  - Multi-valued attributes are ignored until Step 6.

Step 2: Weak Entity Types

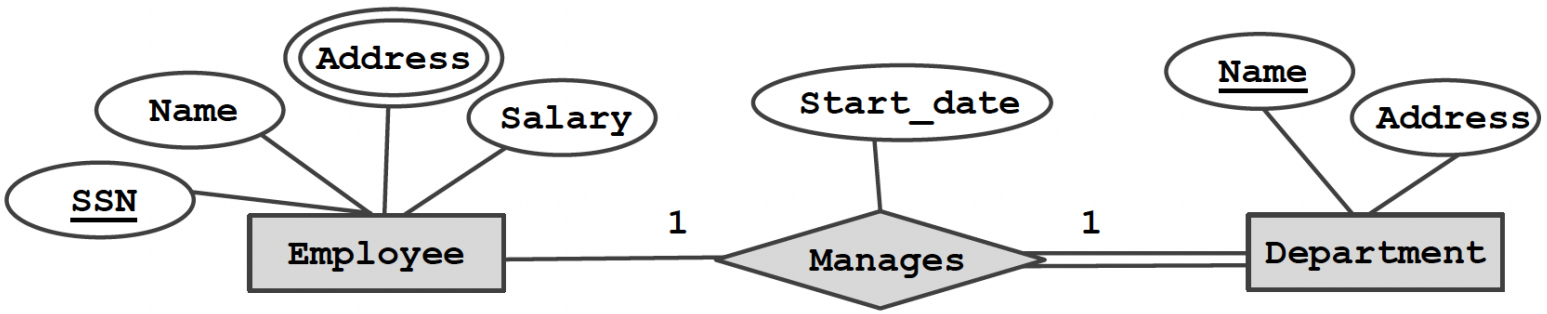
- For each weak entity type E_w, **create a relation schema** with the attributes of E_w plus the PK of its identifying entity type, where
  - **PK:** the partial key attributes of E_w plus the PK of its identifying entity type
  - **FK:** references the PK of its identifying entity type



- CHILD(SSN, Cname, Age) with
  PK: {SSN, Cname}
  FK: [SSN]⊆EMPLOYEE[SSN]

Step 3: Binary 1:1 Relationship Types - (Foreign key approach)

- For a 1:1 relationship type *R* with one total participation, **extend the relation schema of the total-side entity type** by the attributes of *R* and the PK of the partial-side entity type, where
  - **PK:** still the PK of the total-side entity type
  - **FK:** references the PK of the partial-side entity type



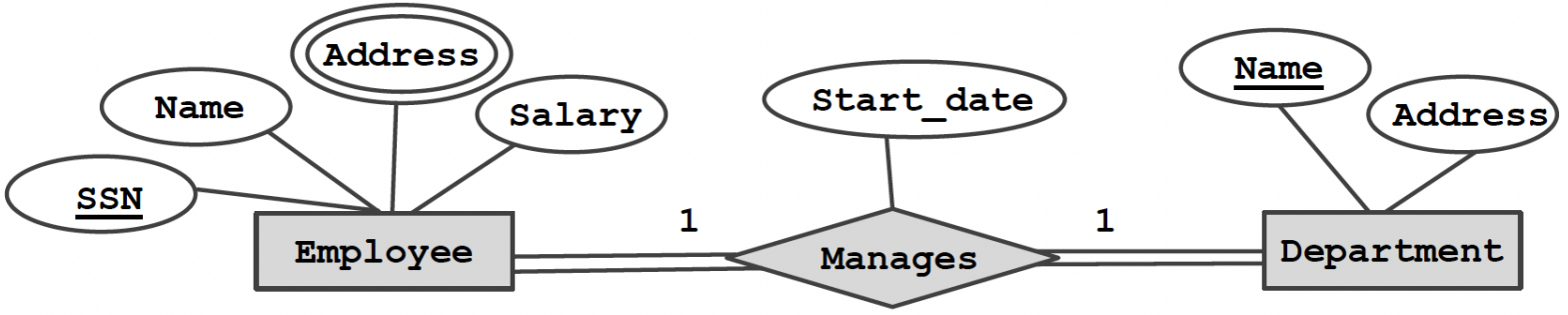
- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

  PK: {Name}

  FK: [Mgr_SSN]⊆EMPLOYEE[SSN].

Step 3: Binary 1:1 Relationship Types - (Merged relation approach)

- How can we translate the following kind of 1:1 relationship type?

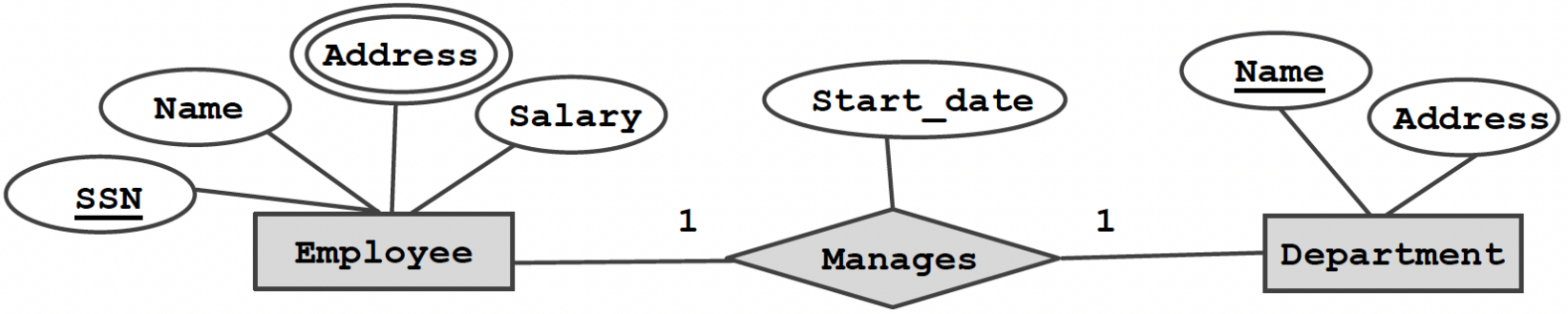


- If participation on both sides is total, we may **merge the relation schemas of both entity types and the attributes of the relationship type into a single relation**.

- EMPLOYEE-DEP(SSN, Name, Salary, Start_date, Dname, Address) with PK: {SSN} or {Dname}

Step 3: Binary 1:1 Relationship Types - (Cross-reference approach)

- How can we translate the following kind of 1:1 relationship type?



- If both sides are partial, we may **create a relation schema** which cross-references the PKs of the relation schemas of the two entity types.

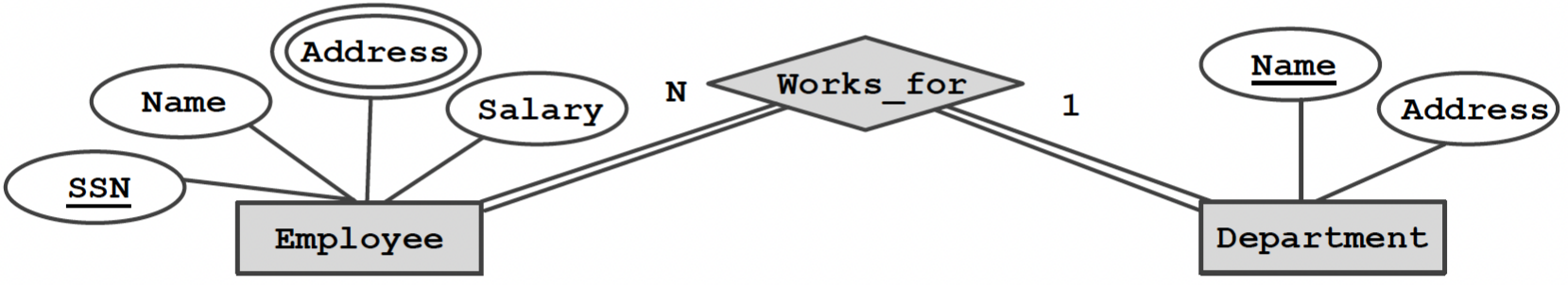
- MANAGES(SSN, Dname, Start_date) with

  PK: {SSN} or {Dname}

  FKs: [SSN]⊆EMPLOYEE[SSN] and [Dname]⊆DEPARTMENT[Name]

Step 4: Binary 1:N Relationship Types

- For each 1:N relationship type *R*, **extend the relation schema of the N-side entity type** by the attributes of *R* and the PK of the 1-side entity type, where
  - **PK:** still the PK of the N-side entity type
  - **FK:** references the PK of the 1-side entity type



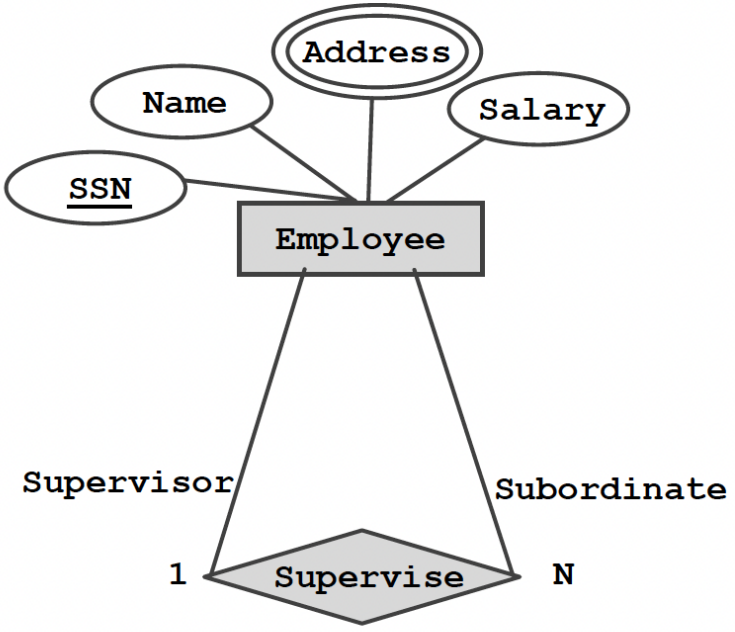
- EMPLOYEE(SSN, Name, Salary, Dname) with

  PK: {SSN}

  FK: [Dname]⊆DEPARTMENT[Name]

Step 4: Binary 1:N Relationship Types

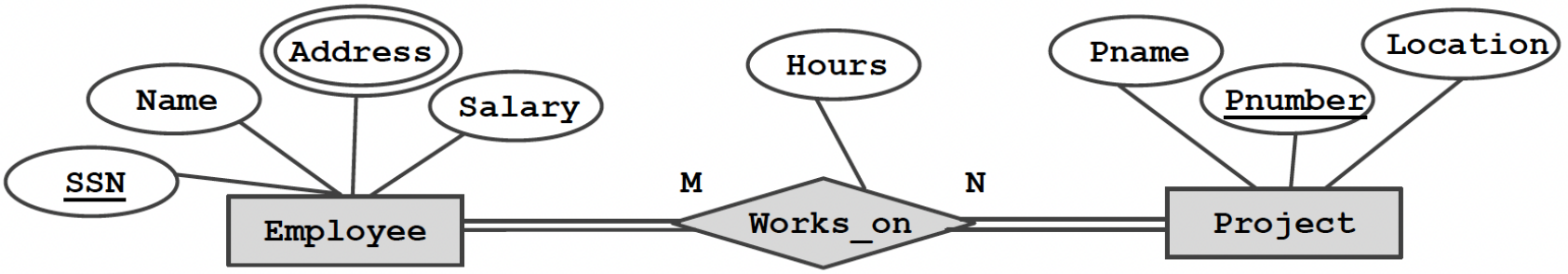
- How can we translate the 1:N relationship type SUPERVISE?



- EMPLOYEE(SSN, Name, Salary, Dname, Super_SSN) with
  PK: {SSN}
  FK: [Dname]⊆DEPARTMENT[Name] and [Super_SSN]⊆EMPLOYEE[SSN]

Step 5: Binary M:N Relationship Types

- For each M:N relationship type *R*, **create a relation schema** with the attributes of *R* plus the PKs of the participating entity types, where
  - **PK:** the combination of the PKs of the participating entity types
  - **FKs:** references the PKs of the participating entity types



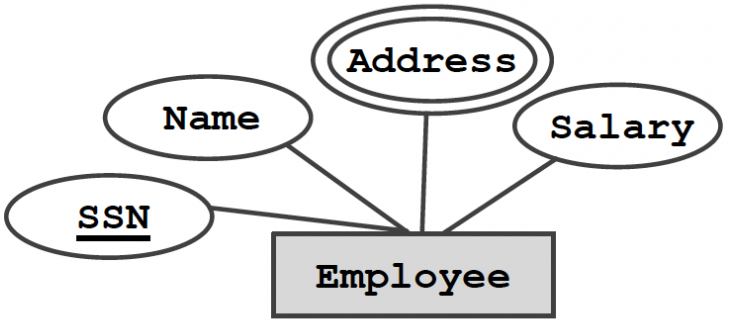
- WORKS_ON(SSN, Pnumber, Hours) with

  PK: {SSN, Pnumber}

  FKs: [SSN]⊆EMPLOYEE[SSN] and [Pnumber]⊆PROJECT[Pnumber]

Step 6: Multi-valued Attributes

- For each multi-valued attribute *A*, **create a relation schema** with an attribute corresponding to A plus the PK of the entity/relationship type that has *A* as an attribute, where
  - **PK:** the combination of *A* and the PK of the entity/relationship type that has *A*
  - **FK:** references the PK of the entity/relationship type that has *A*



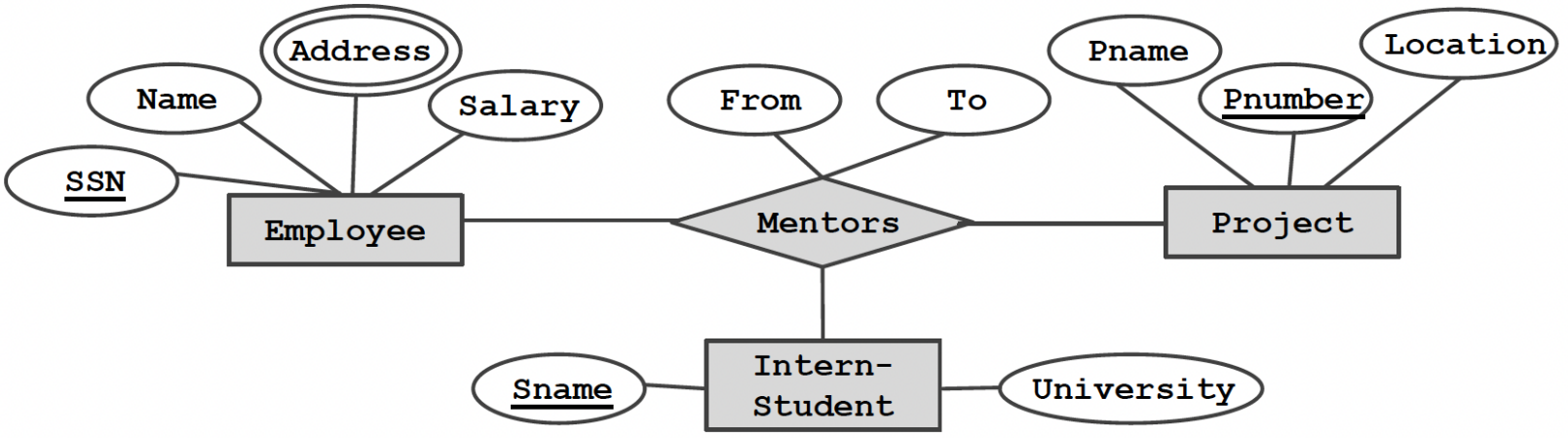
- EMPLOYEE_ADDRESS(SSN, Address) with

  PK: {SSN, Address}

  FK: [SSN]$\subseteq$EMPLOYEE[SSN]

Step 7: N-ary Relationship Types

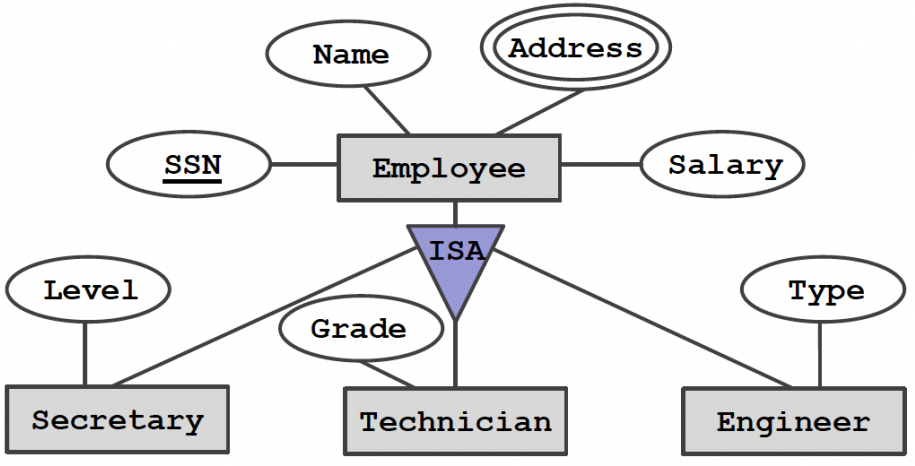
- For each N-ary relationship type *R*, **create a relation schema** with the attributes of *R* plus the PKs of the participating entity types, where
  - **PK:** the combination of the PKs of the participating entity types
  - **FKs:** references the PKs of the participating entity types



- MENTORS(SSN, Sname, Pnumber, From, To) with
  PK: {SSN, Sname, Pnumber}
  FK: [SSN]⊆EMPLOYEE[SSN], [Sname]⊆INTERN_STUDENT[Sname], and
  [Pnumber]⊂PROJECT[Pnumber]

Step 8: Superclass and Subclass

- For each superclass, **create a relation schema** with its attributes.
- For each subclass, **create a relation schema** with its attributes plus the key attributes of its superclass.
  - **PK:** the PK of the superclass
  - **FK:** references the PK of the superclass



- EMPLOYEE(...) (as done before)
- SECRETARY(SSN, Level), TECHNICIAN(SSN, Grade), ENGINEER(SSN, Type), which all have

  PK: {SSN}

  FK: [SSN]$\subseteq$EMPLOYEE[SSN]

ER-to-Relations Algorithm (Recall)

- The algorithm to first convert the basic ER model into relations, and then convert superclass/subclass from the EER model into relations.

  Step 1: Mapping of Regular Entity Types

  Step 2: Mapping of Weak Entity Types

  Step 3: Mapping of Binary 1:1 Relationship Types

  - Foreign key approach
  - Merged relation approach
  - Cross-reference approach

  Step 4: Mapping of Binary 1:N Relationship Types

  Step 5: Mapping of Binary M:N Relationship Types

  Step 6: Mapping of Multi-valued Attributes

  Step 7: Mapping of N-ary Relationship Types

  Step 8: Mapping of Superclass/Subclass

Australian National University

A Relational Database Schema - The Company Database

- EMPLOYEE(SSN , Name, Salary, Dname Super_SSN)
- WORKS_ON(SSN , Pnumber , Hours)
- DEPARTMENT(Name , Address, Mgr_SSN , Start_date)
- PROJECT(Pnumber , Pname, Location, Dname)
- EMPLOYEE_ADDRESS(SSN , Address)
- CHILD(SSN , Cname, Age)