Australian National University

Entity-Relationship Model – Part 1

Database Design Process

IT Projects[1]

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

[1] CHAOS report by Standish Group, 2015

IT Projects[1]

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

- There can be many reasons, including:
  - Customers were not sure about what they wanted,
  - Requirements were not properly documented,
  - Improper development methodology was used,
  - Resources were not sufficient,
  - There were communication issues,
  - . . .

[1] CHAOS report by Standish Group, 2015

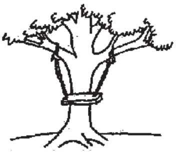
The Tyre Swing Project

As proposed by the project sponsor

As specified in the project request

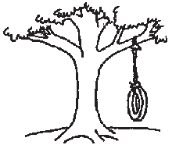
As designed by the senior analyst

As produced by the programmers



As installed at the user's site

What the user wanted



Database Design – Four Phases

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

  1. **Requirements Collection and Analysis**
  2. **Conceptual Design**
  3. **Logical Design**
  4. **Physical Design**

Phase 1: Requirements Collection and Analysis

- **Requirements collection and analysis** is the process of collecting and analyzing data requirements of the organization so as to provide database solutions that fulfill business needs of the organization.
- Compilation of data requirements includes:
  - a description of data used or generated;
  - details of how data is to be used/generated;
  - any additional requirements for new database system;
  - . . .

Phase 2: Conceptual Design

- **Conceptual design** is the process of constructing a conceptual data model that is
  - modeled at a high-level of abstraction;
  - sufficiently simple and often graphical;
  - used to communicate the requirements of a database with nontechnical users.
- A conceptual data model is built using the information in users' requirements specification.

  Note: The conceptual design is based on **the entity-relationship model** in this course.

Phase 3: Logical Design

- **Logical design** is the process of constructing a logical data model (e.g. relational or object-oriented).
- A conceptual data model is translated onto a logical data model, which can be further refined (e.g., normalisation) to meet the data requirements. For example,
    - **From:** An ER model
    - **To:** Relations with their primary and foreign keys, which facilitates SQL to deal with retrieving, updating and deletion.

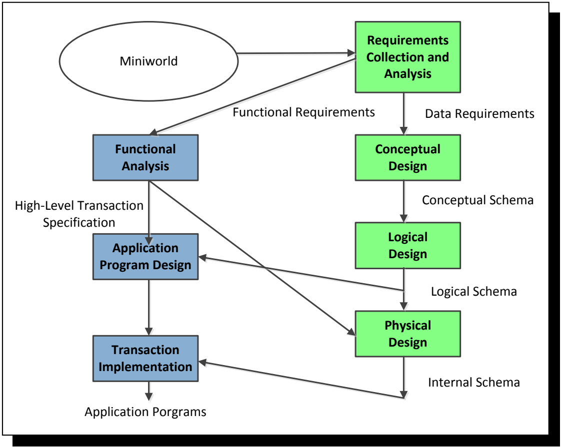
Note: The logical design is based on **the relational data model** in this course.

Phase 4: Physical Design

- **Physical design** is the process of implementing the logical data model in a specific database management system (DBMS).
- Assume that the logical data model is the relational data model. Then the physical design is to create relations in a DBMS that involves:
  - Selecting the files in which to store the relations.
  - Deciding which indexes should be used to achieve efficient access.
  - Describing the integrity constraints and security measures.
  - . . .
- The decisions made during the physical design phrase affect the performance and accessibility of the database.

  Note: Details of this topic are out of the scope of our course.

Database Design Process



Entity-Relationship Model – Part 2

Basic Modeling Concepts

Entity-Relationship (ER) Model

- Originally proposed by Peter Chen in 1976.
  - Shortly after its introduction, the ER model became the most popular data model used in conceptual database design.

Entity-Relationship (ER) Model

- Originally proposed by Peter Chen in 1976.
  - Shortly after its introduction, the ER model became the most popular data model used in conceptual database design.
- A data model normally has three key aspects:

  (1) **Data structure**:

  Data in the ER model is represented as **entities** and **relationships** with **attributes**.

  (2) **Data integrity**:

  For the ER model, **keys** are for entity/relationship types, and **cardinality/participation constraints** for relationship types.

  (3) **Data manipulation**:

  No standard data manipulation operations are associated with the ER model.

Entity-Relationship (ER) Model

- Comparing key concepts in the relational data model and the ER model:

<table>
<tr><th>Relational Data Model</th><th>Entity-Relationship Model</th></tr>
<tr><td colspan="2">Attribute</td></tr>
<tr><td colspan="2">Domain</td></tr>
<tr><td colspan="2">Superkey/primary key/candidate key</td></tr>
<tr><td>Tuple</td><td>Entity/Relationship</td></tr>
<tr><td>Relation</td><td>Entity set/Relationship set</td></tr>
<tr><td>Relation schema</td><td>Entity type/Relationship type</td></tr>
</table>

Entity-Relationship (ER) Model

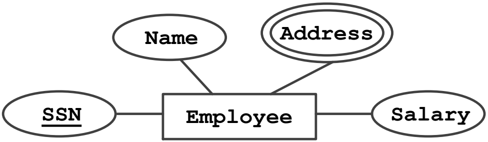
- **ER diagrams:** diagrammatic notation associated with the ER model.
  - They are relatively simple;
  - They are user-friendly;
  - They can provide a unified view of data, which is independent of any implemented data model.
- There are a number of ER diagrammatic notations available. We shall closely follow the one used by Chen and its variations.
  - **Attributes** are represented as *ovals*;
  - **Key attributes** are *underlined*;
  - **Entity types** are represented as *rectangles*;
  - **Relationship types** are represented as *diamonds*.

Entities and Attributes

- **Entities:** “Things” in the real world (with independent existence).
  - e.g., an individual person
- **Relationships:** Associations between entities.
  - e.g., a person is a friend of another person
- **Attributes:** Properties that describe entities and relationships.
  - **Composite** versus **simple** (atomic) attributes
  - **Single-valued** versus **multivalued** attributes
  - **Stored** versus **derived** attributes
  - **NULL** values
  - **Complex** (nesting of composite and multivalued) attributes
- **Domains of attributes:** For each attribute, a domain is associated, i.e., a set of permitted values for an attribute.

Entity Types and Entity Sets

- An **entity type** defines a collection (or set) of entities that have the same attributes.
  - Described by its name and attributes.
- An **entity set** is a collection of all entities of a particular entity type in the database at any point in time.



Relationship Types and Relationship Sets

- A **relationship type** is an association between two or more entity types, and can have attributes as well.
  (We also say: such entity types **participate in** a relationship type)

  Example:

  - **Employee** works-for **Department**
  - **Employee** registers a **Customer** at **Branch office**

- **Degree of relationship type:** the number of participating entity types. We can have binary, ternary,. . . ,nary.
- A **relationship set** is the set of associations between entities of the entity types that participate in the relationship type.

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.

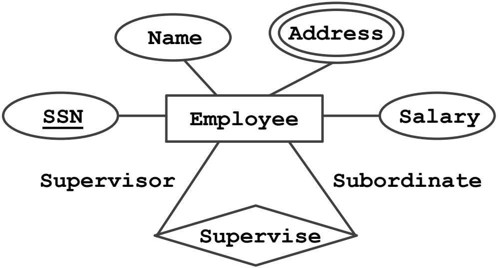
Constraints on Relationships - Cardinality Limits

- Instead of cardinality ratios or participation constraints, more precise **cardinality limits** can be associated with relationship types.
- Each entity type participating in a relationship type associates with a pair of integer numbers **(min, max)**.

Meaning: An employee must work for exactly one department and each department must have one or more employees.

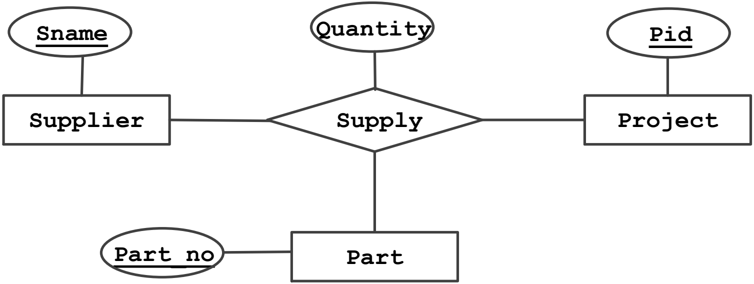
Recursive Relationships

- **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.



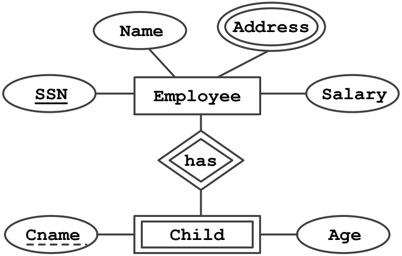
Higher-Degree Relationship Types

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



Weak Entity Types

- A **weak entity type** is an entity type that does not have sufficient attributes to form a primary key.
  - Its existence depends on the existence of an identifying entity type, and the relationship between them is called an **identifying relationship**.
  - It must have one or more attributes, together with the primary key of the identifying entity type, for distinguishing its entities.



Design Choices for the ER Model

- It is possible to define entities and their relationships in a number of different ways.
- Some questions:
  - Should a concept be modeled as **an entity type or an attribute**?
  - Should a concept be modeled as **an entity type or a relationship type**?
  - Should a concept be modeled as **a ternary relationship type or several binary relationship types**?

Entity-Relationship Model – Part 3

Enhanced Modeling Concepts

Enhanced Entity-Relationship (EER) Model

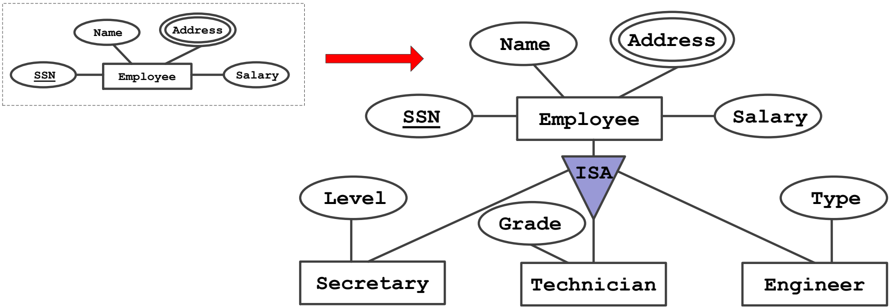
- The basic modelling concepts are only sufficient for some database applications.
- To reflect data properties and constraints more precisely, a number of enhanced ER models (EERs) were proposed.
- Each EER model includes all the basic modeling concepts of the ER model we discussed before.
- We will 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**.

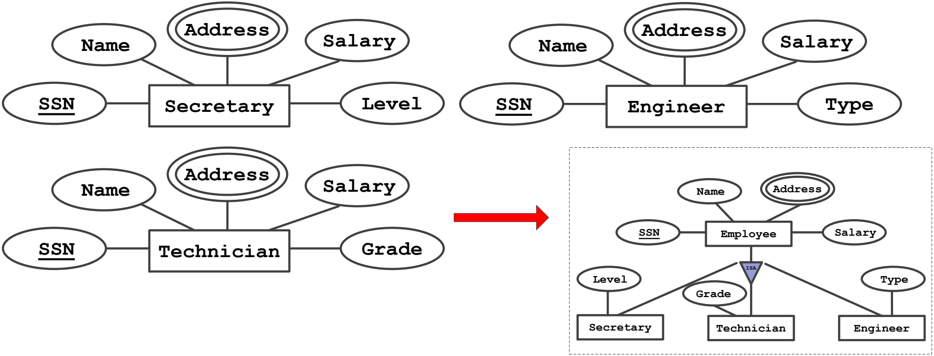
Specialisation and Generalisation

- **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:



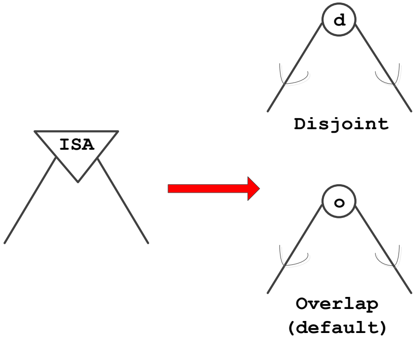
Specialisation and Generalisation

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



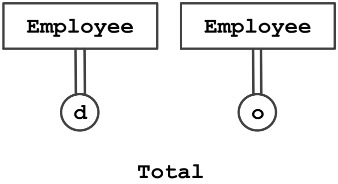
Constraints on Specialisation and Generalisation

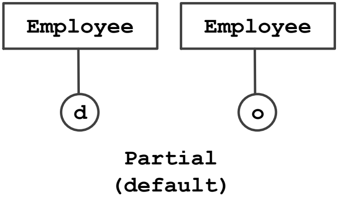
- **Disjointness constraint**
  - Specifies that the subclasses of the specialization must be **disjoint**.
  - If not constrained, then entities in the subclasses may **overlap**.



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.





Design Choices for the EER Model

- Specializations and generalisation can be defined to make the conceptual model accurate.
- If the subclasses has few specific attributes and no specific relationships, then
  - can be merged into the superclass,
  - replace with one or more type attributes specifying the subclass that each entity belongs to.
- Choices of disjoint/overlapping and total/partial constraints are driven by rules in the miniworld being modeled.

Informal Method for Constructing 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 (including ISA and identifying relationship types)

  (3) Identify the attributes of entity and relationship types (and their underlying domains)

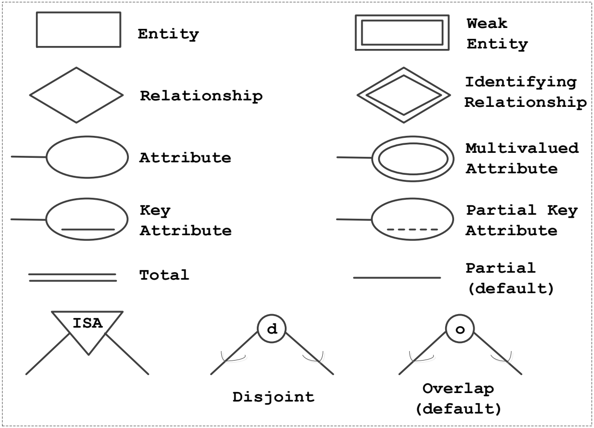
  (4) Identify a primary key for each entity type

  (5) Classify each binary relationship type identified in step 2 (i.e. one-to-one, many-to-one or many-to-many)

  (6) Determine the participation constraints for each entity type in each binary relationship type

  (7) Determine the disjointness and completeness constraints for each ISA

Summary of Notation for ER and EER Diagrams



Entity-Relationship Model – Part 4

From ER to Relations

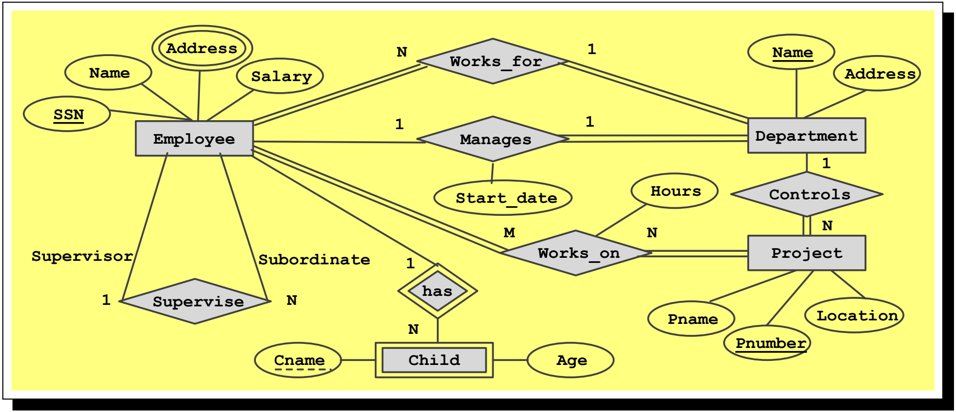
Recap - Data Modeling

Requirements → ER diagram → Relational database schema → Relational DBMS

Conceptual level | Logical level | Physical level

- ER design is **subjective**:
  - There are many ways to model a given scenario.
  - Analyzing alternative schemas is important.
- Constraints play an important role in designing a good database. But,
  - Not all constraints can be expressed in the ER model;
  - Not all constraints in the ER model can be translated.
- A good database design requires to further refining a relational database schema obtained through translating an ER diagram.

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.

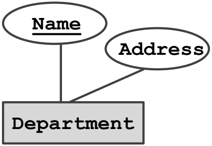
  Step 1: Mapping of Regular Entity Types
  Step 2: Mapping of Weak Entity Types
  Step 3: Mapping of Binary 1:1 Relationship Types

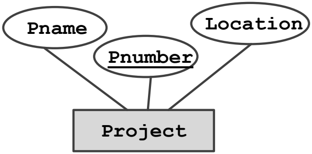
  - Foreign key approach
  - Merged relation approach
  - Cross-reference approach

  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*

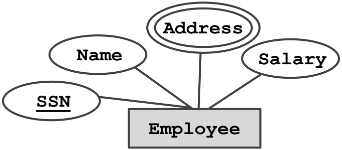




- DEPARTMENT(Name, Address) with PK: {Name}
  PROJECT(Pnumber, Pname, Location) with PK: {Pnumber}
- **Note:** These are not necessarily the final relation schemas of DEPARTMENT and PROJECT.

Step 1: Regular Entity types

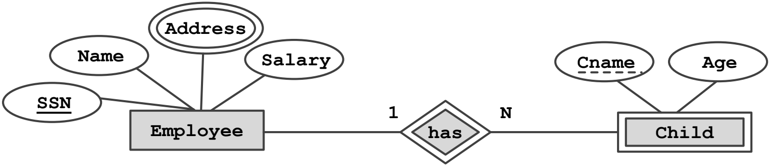
- How can we translate the regular entity type EMPLOYEE?



- 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

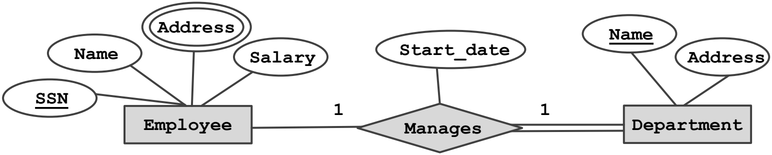
- For each weak entity type E_w, **create a relation schema** with the attributes of E_w plus the PK of its identifying entity type, where
  - **PK:** the partial key attributes of E_w plus the PK of its identifying entity type
  - **FK:** references the PK of its identifying entity type



- CHILD(SSN, Cname, Age) with
  PK: {SSN, Cname}
  FK: [SSN]$\subseteq$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



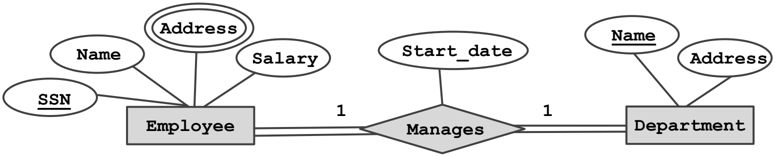
- DEPARTMENT(Name, Address, Mgr_SSN, Start_date) with

  PK: {Name}

  FK: [Mgr_SSN]⊆EMPLOYEE[SSN].

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

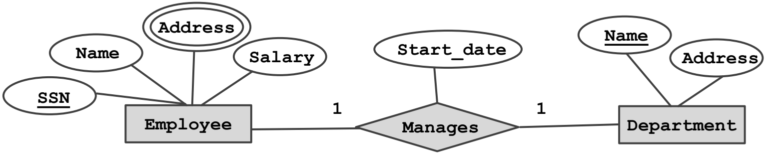
- How can we translate the following kind of 1:1 relationship type?



- 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?

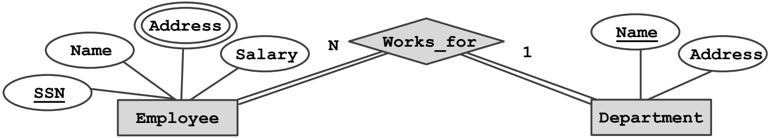


- If both sides are partial, we may **create a relation schema** which cross-references the PKs of the relation schemas of the two entity types.

- MANAGES(SSN, Dname, Start_date) with
  PK: {SSN} or {Dname}
  FKs: [SSN]$\subseteq$EMPLOYEE[SSN] and [Dname]$\subseteq$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



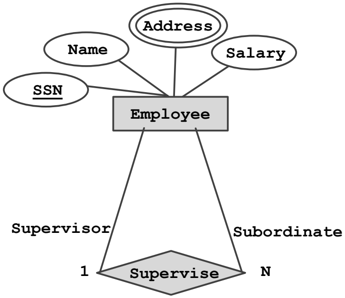
- EMPLOYEE(SSN, Name, Salary, Dname) with

  PK: {SSN}

  FK: [Dname]$\subseteq$DEPARTMENT[Name]

Step 4: Binary 1:N Relationship Types

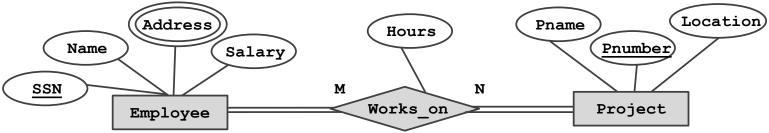
- How can we translate the 1:N relationship type SUPERVISE?



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



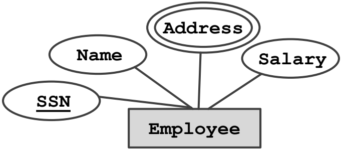
- WORKS_ON(SSN, Pnumber, Hours) with

  PK: {SSN, Pnumber}

  FKs: [SSN]$\subseteq$EMPLOYEE[SSN] and [Pnumber]$\subseteq$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*



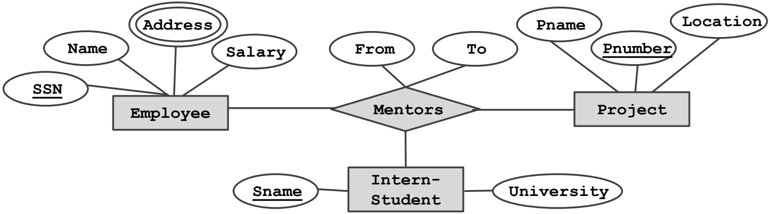
- EMPLOYEE_ADDRESS(SSN, Address) with

  PK: {SSN, Address}

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

Step 7: N-ary Relationship Types

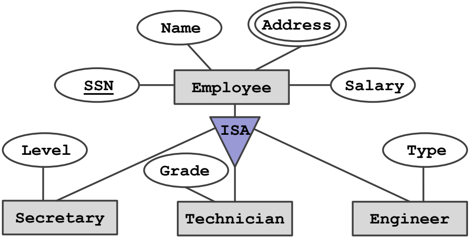
- For each N-ary relationship type *R*, **create a relation schema** with the attributes of *R* plus the PKs of the participating entity types, where
  - **PK:** the combination of the PKs of the participating entity types
  - **FKs:** references the PKs of the participating entity types



- 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]⊆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

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)