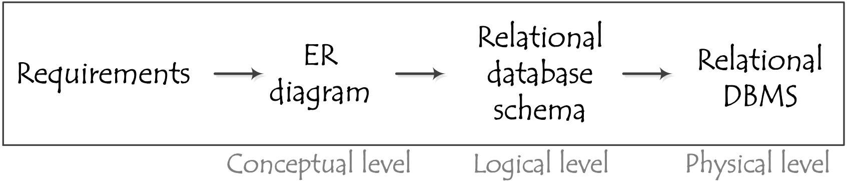
Entity-Relationship Model – Part 4

From ER to Relations

Recap - Data Modeling



Requirements → ER diagram → Relational database schema → Relational DBMS

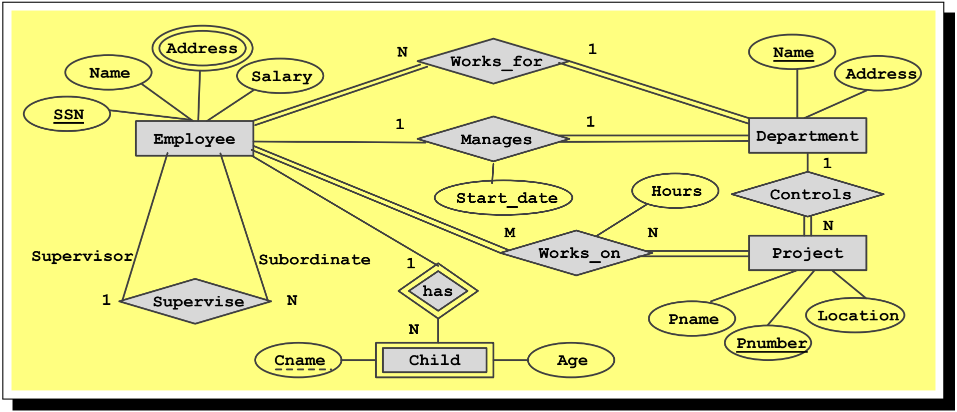
Conceptual level

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.

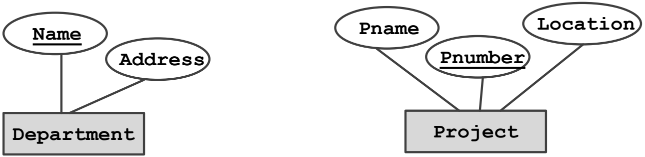
Step 1: Mapping of Regular Entity Types
Step 2: Mapping of Weak Entity Types
Step 3: Mapping of Binary 1:1 Relationship Types

  - 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

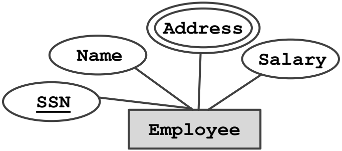
- For each regular entity type *E*, **create a relation schema** with the attributes of *E* (ignore multi-valued attributes until Step 6), where
  - **PK:** the key attributes of *E*



- 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

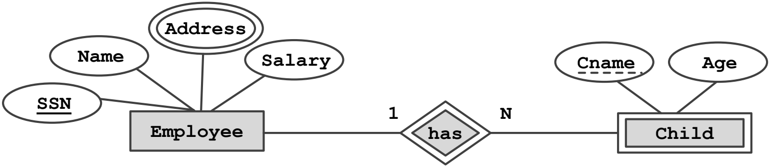
- How can we translate the regular entity type EMPLOYEE?



- 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

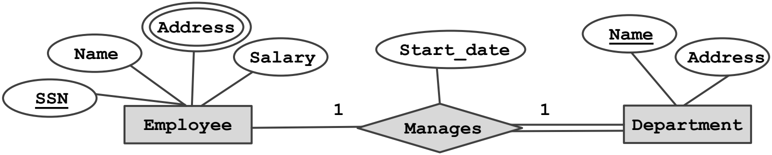
- For each weak entity type E_w, **create a relation schema** with the attributes of E_w plus the PK of its identifying entity type, where
  - **PK:** the partial key attributes of E_w plus the PK of its identifying entity type
  - **FK:** references the PK of its identifying entity type



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



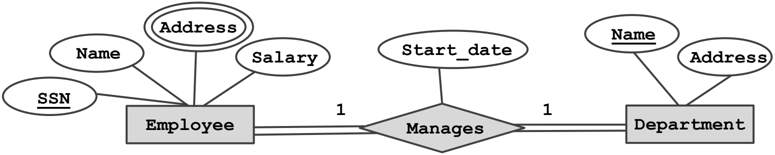
- Department(Name, Address, Mgr_SSN, Start_date) with

  PK: {Name}

  FK: [Mgr_SSN]⊆Employee[SSN].

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

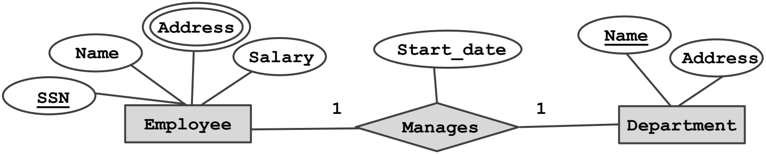
- How can we translate the following kind of 1:1 relationship type?



- 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.

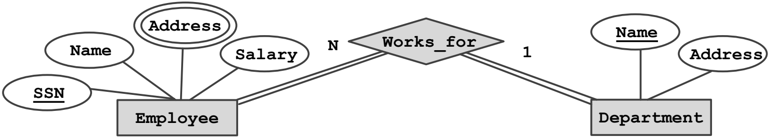
- MANAGES(SSN, Dname, Start_date) with

  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



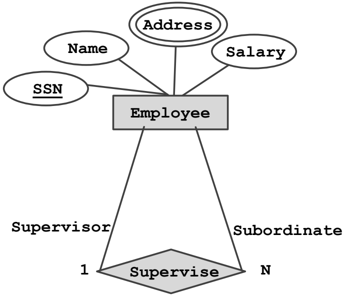
- EMPLOYEE(SSN, Name, Salary, Dname) with

  PK: {SSN}

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

Step 4: Binary 1:N Relationship Types

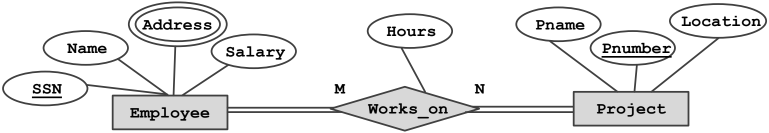
- How can we translate the 1:N relationship type SUPERVISE?



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



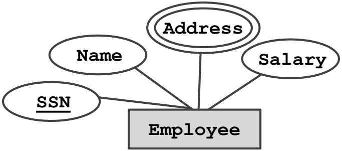
- WORKS_ON(SSN, Pnumber, Hours) with

  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*



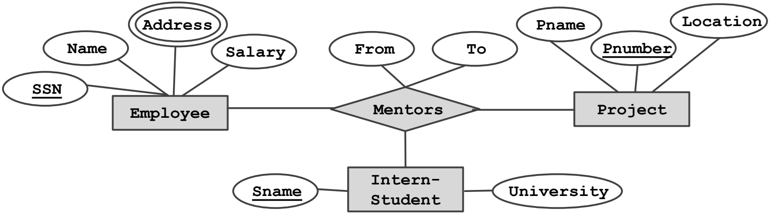
- EMPLOYEE_ADDRESS(SSN, Address) with

  PK: {SSN, Address}

  FK: [SSN]⊆EMPLOYEE[SSN]

Step 7: N-ary Relationship Types

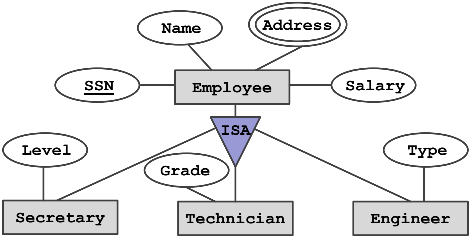
- For each N-ary relationship type *R*, **create a relation schema** with the attributes of *R* plus the PKs of the participating entity types, where
    - **PK:** the combination of the PKs of the participating entity types
    - **FKs:** references the PKs of the participating entity types



- 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.

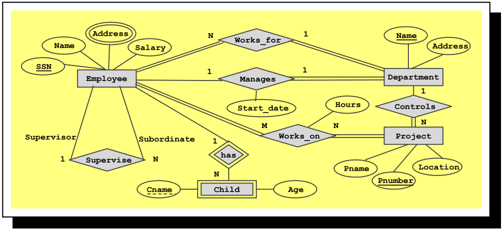
  Step 1: Mapping of Regular Entity Types
  Step 2: Mapping of Weak Entity Types
  Step 3: Mapping of Binary 1:1 Relationship Types

  - 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

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)


Address
Name
Salary
SSN
Employee
N
Works_for
1
Name
Address
Department
1
Manages
1
Start_date
Hours
1
Controls
N
M
Works_on
N
Project
Supervisor
Subordinate
1
has
1
Supervise
N
N
Pname
Location
Pnumber
Cname
Child
Age