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:

Relational Data Model	Entity-Relationship Model
Attribute	
Domain	
Superkey/primary key/candidate key	
Tuple	Entity/Relationship
Relation	Entity set/Relationship set
Relation schema	Entity type/Relationship type

Entity-Relationship (ER) Model

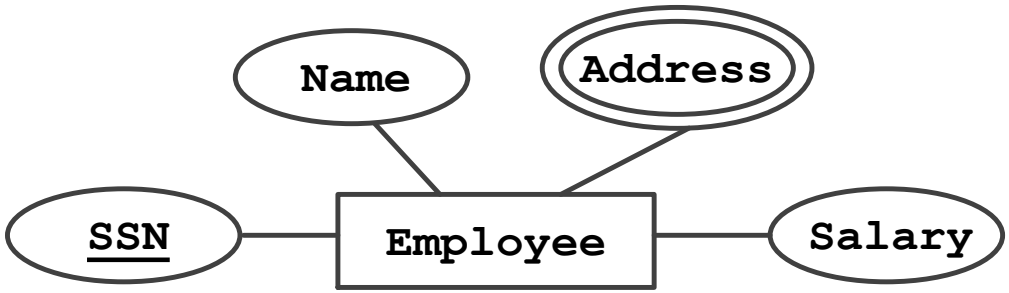
- **ER diagrams:** diagrammatic notation associated with the ER model.
  - They are relatively simple;
  - They are user-friendly;
  - They can provide a unified view of data, which is independent of any implemented data model.
- There are a number of ER diagrammatic notations available. We shall closely 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.

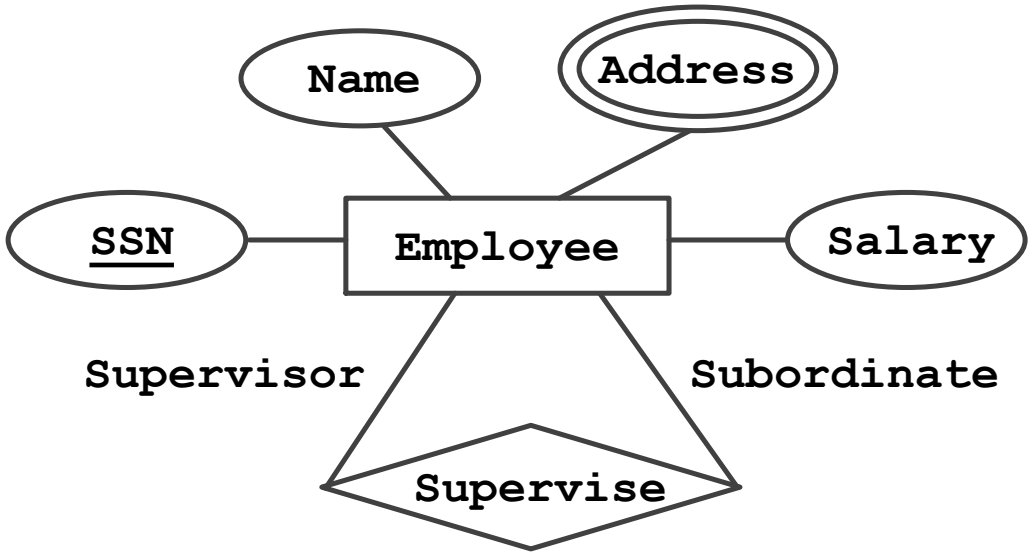
Constraints on Relationships - Cardinality Limits

- 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.

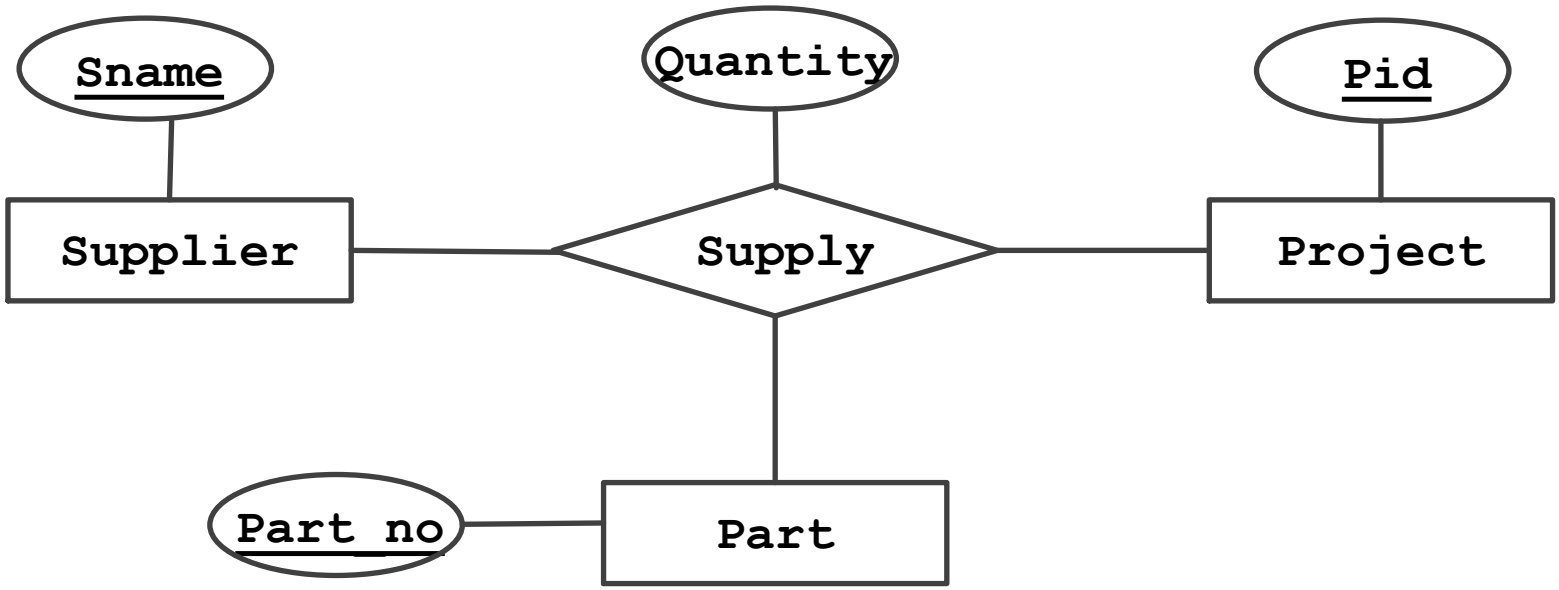
Recursive Relationships

- **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.



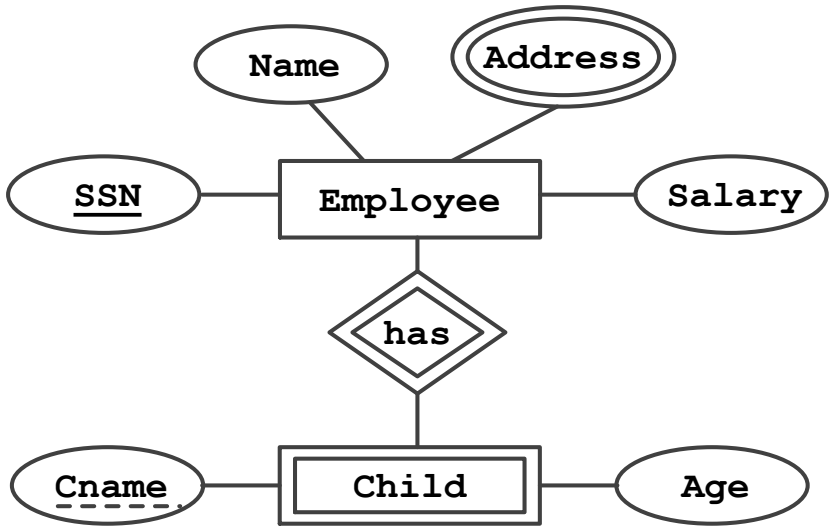
Higher-Degree Relationship Types

- 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**?