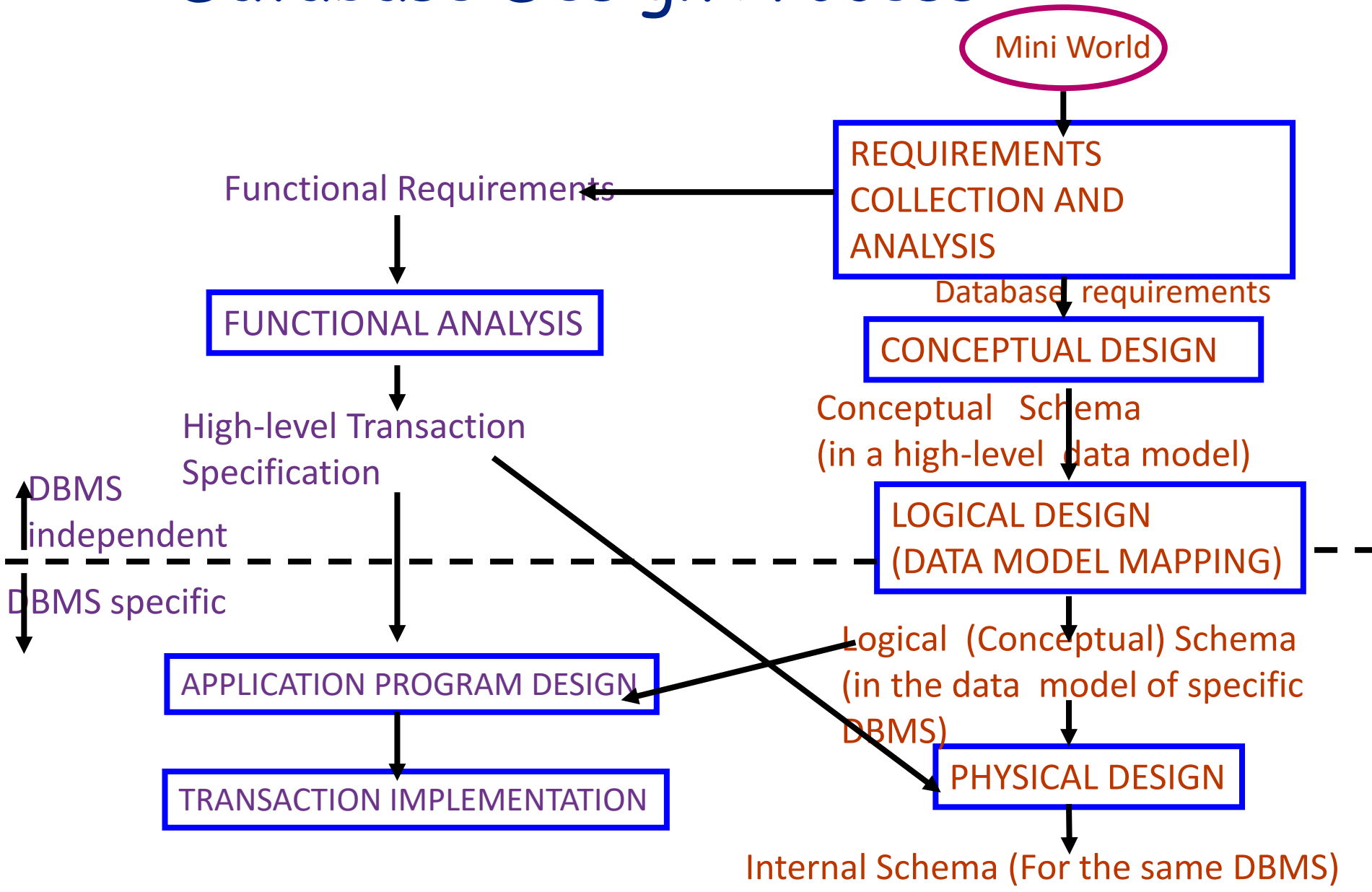
# Entity-Relationship (ER) Data Model

# Database Design Process



# Phases of Database Design

Requirements collection and analysis

- Database Requirements
- Functional Requirements (operations on database)

Conceptual Design & Functional Analysis

- Create a conceptual schema
- High level transaction specification corresponding to operations on database

Logical Design

- Map conceptual database schema to implementation database schema

Physical Design

- Internal storage structures and file organizations are specified

Application Program Design & Transaction Implementation (in parallel to physical design)

# Example: COMPANY Database

Requirements for the Company Database

- The company is organized into Departments. Each department has a name, number, and an employee who manages the department. We keep track of the start date if the department has a manager. A department may have several locations.
- Each department controls a number of Projects. Each project has a name, number, and is located at a single location.

# Example: COMPANY Database

Requirements for the Company Database

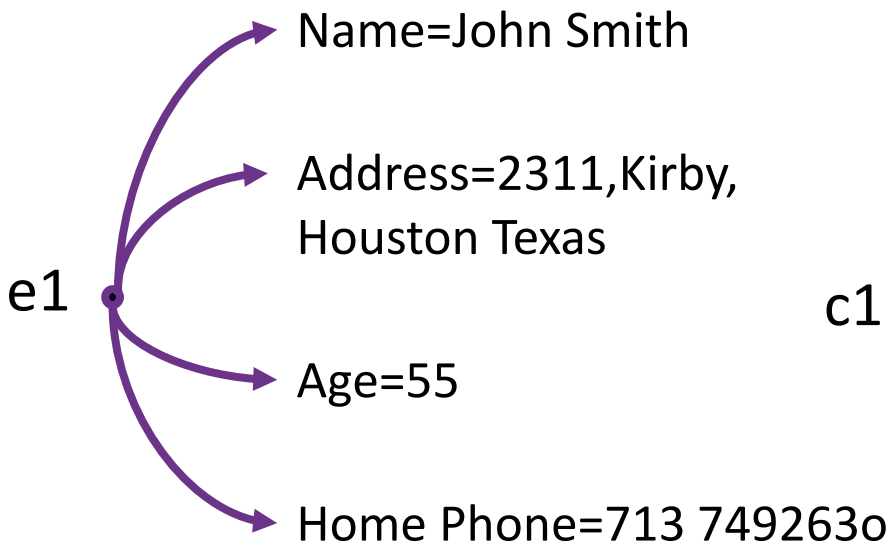
- We store each Employees social security number, name, addresses, salary, sex, and birth date. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.
- Each employee may have a number of Dependents. For each dependent, we keep their name, sex, birthdate, and relationship to the employee.

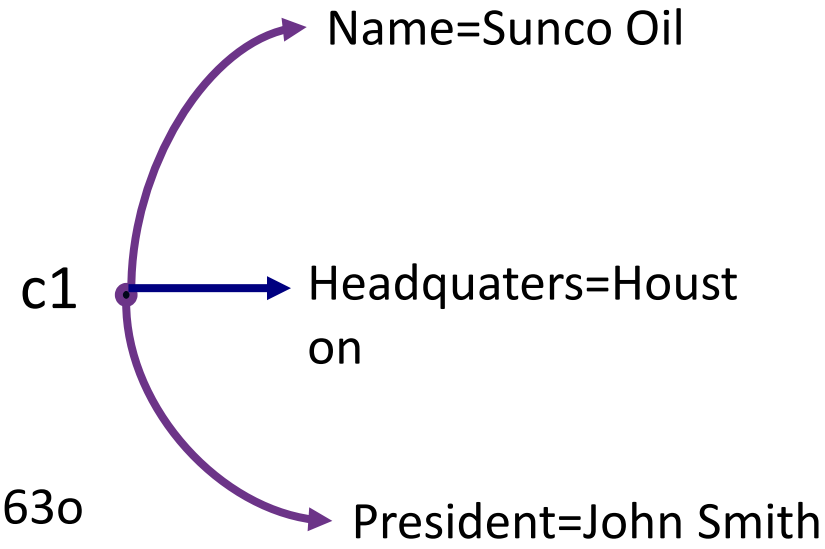
Check last two slides for the ER Model for Company Database

# Entities and Attributes

- Entities are specific objects or things in the mini world that are represented in the database; for example the Employee John Smith, the Research Department, the Database Project.
- Attributes are properties used to describe an entity; for example, an Employee entity may have a Name, ID, Address, Sex, BirthDate.
- A specific entity will have a value for each of its attributes.

# Entities and Attributes



# Entity Types and Key Attributes

Entities with the same basic attributes are grouped or typed into an entity type. For example, the Employee entity type or the Company entity type.

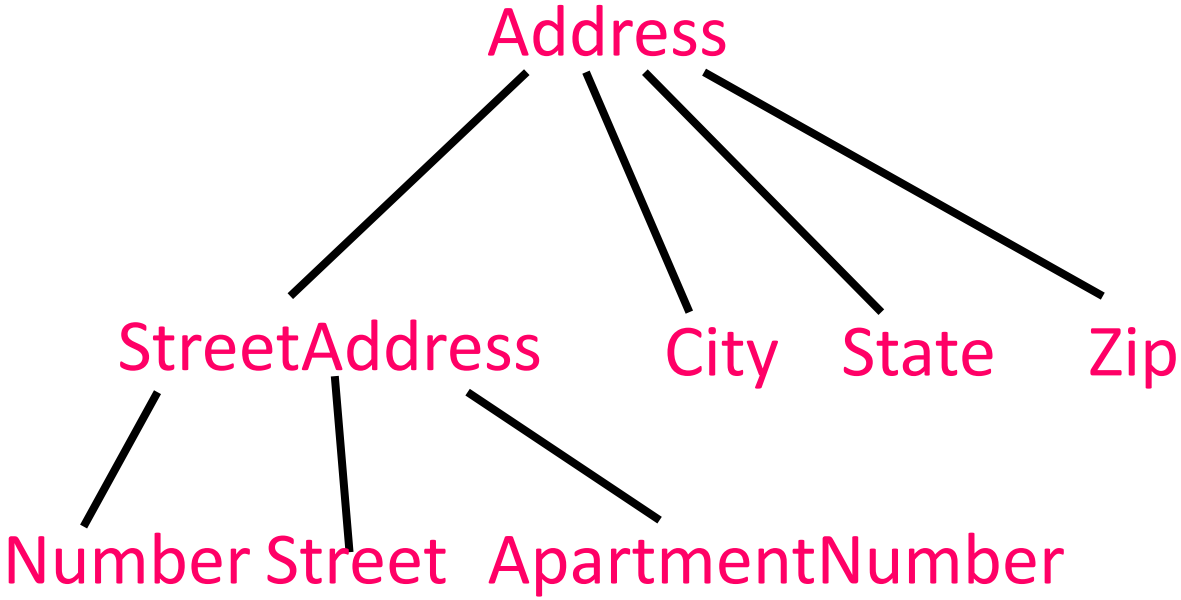
An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type. For example VehicleID of Car Entity

A key attribute can be composite. For example Registration(RegistrationNumber, State) of Car.

An entity type may have more than one key. For example VehicleID and Registration(RegistrationNumber, State) of Car.

# Types of Attributes

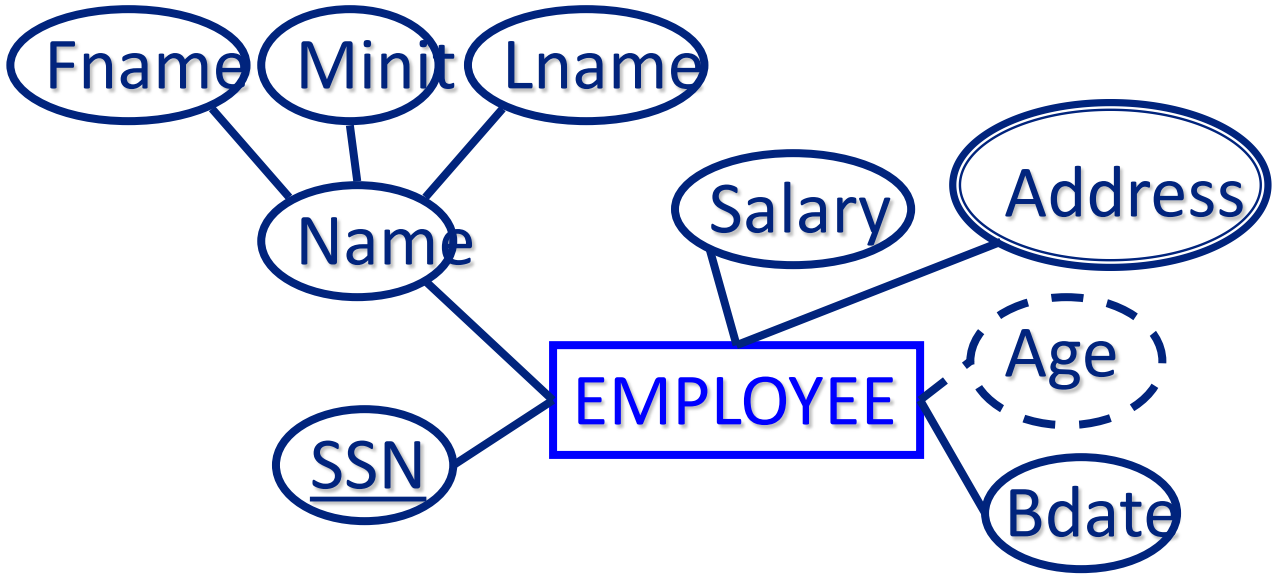
- Simple: Each entity has a single atomic value for the attribute. For example ID, CourseNo.
- Composite: The attribute may be composed of several components. For example Address, StreetAddress.



# Types of Attributes

- Multivalued: An entity may have multiple values for that attribute. For example, Color of a Car or PreviousDegrees of a Ph.D. Student.
- Derived: The domain value of attribute can be determined from one or more other attributes. For example, age from birth date, SGA from course grades.
- In general, composite and multiple-valued attributes may be nested arbitrarily to any number of levels although this is rare.

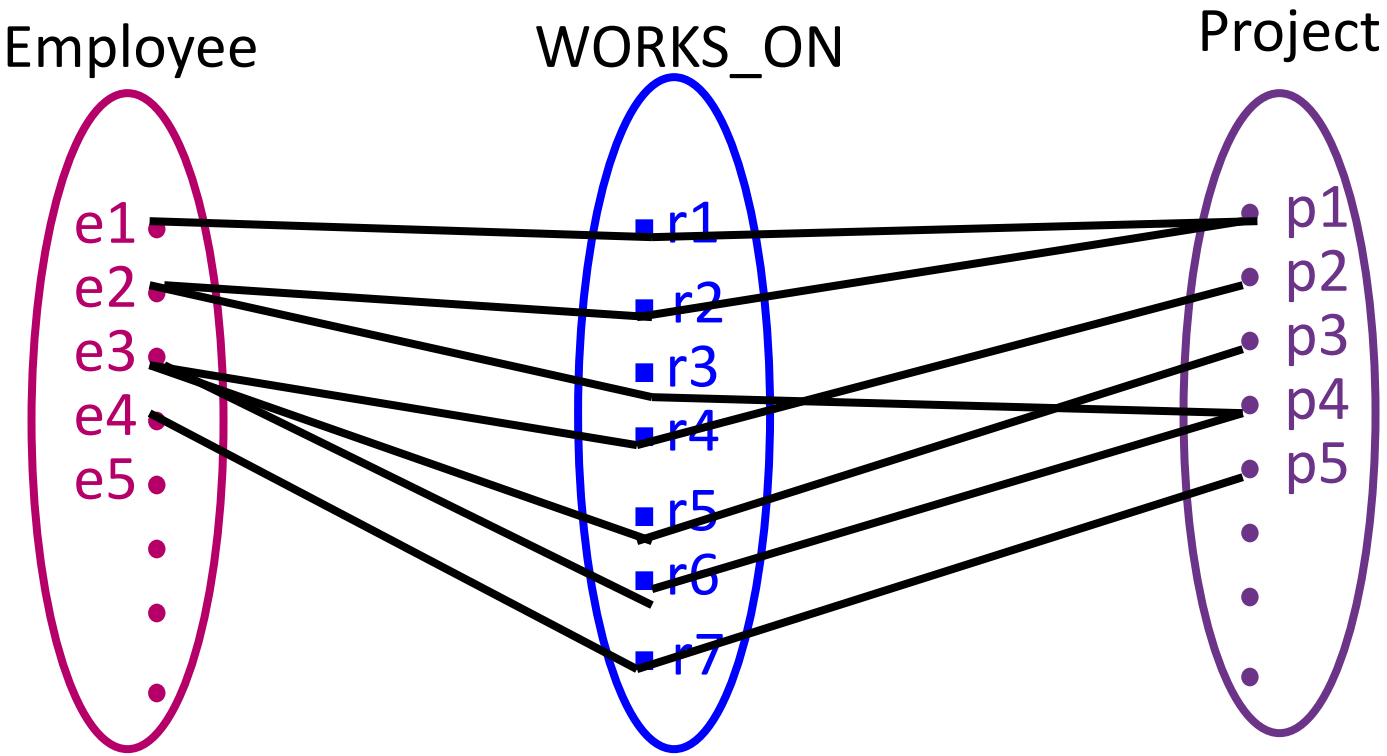
# Visualizing Entity Types & Attributes



# Relationships and Relationship Types

- A Relationship relates two or more distinct entities with a specific meaning. For example, Employee John Smith works on the Database Project, or Employee Jim Wong manages the Research Department.
- Relationships of the same type are grouped or typed into a relationship type. For example, the Works_On relationship type in which Employee's and Project's participate.
- The degree of a relationship type is the number of participating entity types. Works_On is a binary relationship.
- More than one relationship type can exist with same participating entity types. For e.g., Manages & Works_For.

# Relationship Types

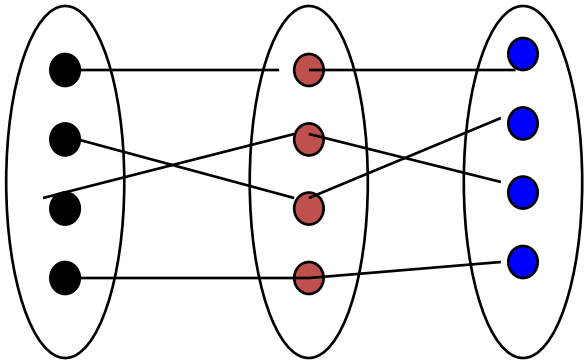



## Attributes for Relationship type

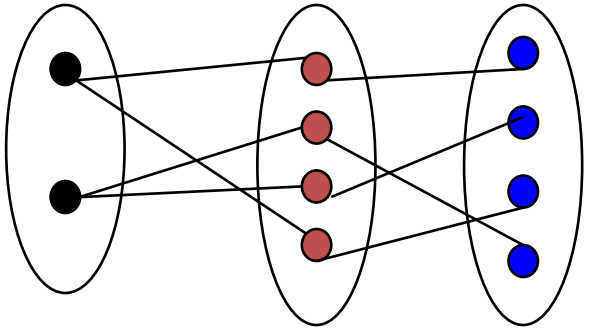
- A relationship type can have attributes; for e.g., HoursPerWeek of Works_On; its value for each relationship instance describes the number of hours per week that an Employee works on a Project

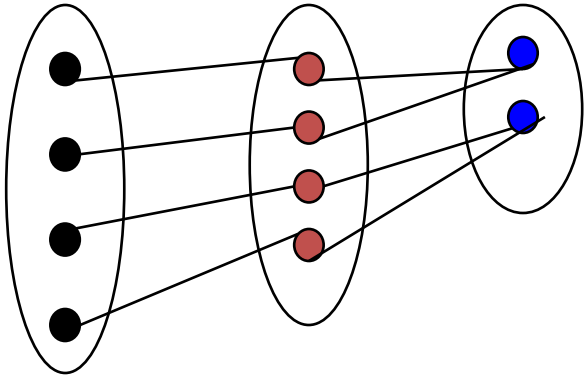
# Relationship Types

One to One

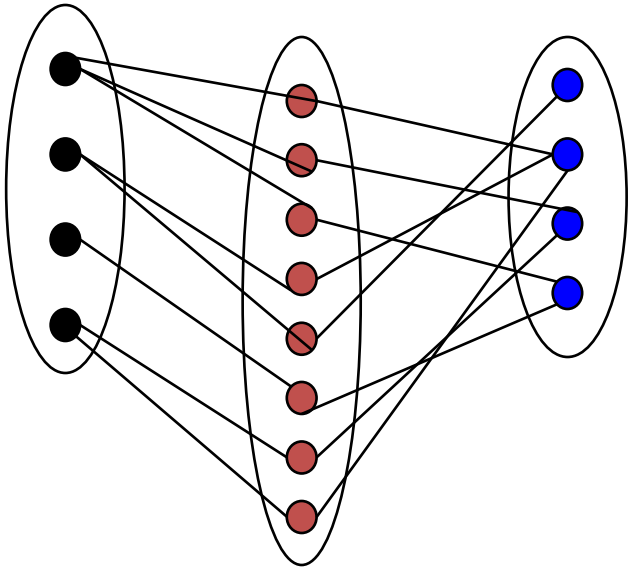


Many to One





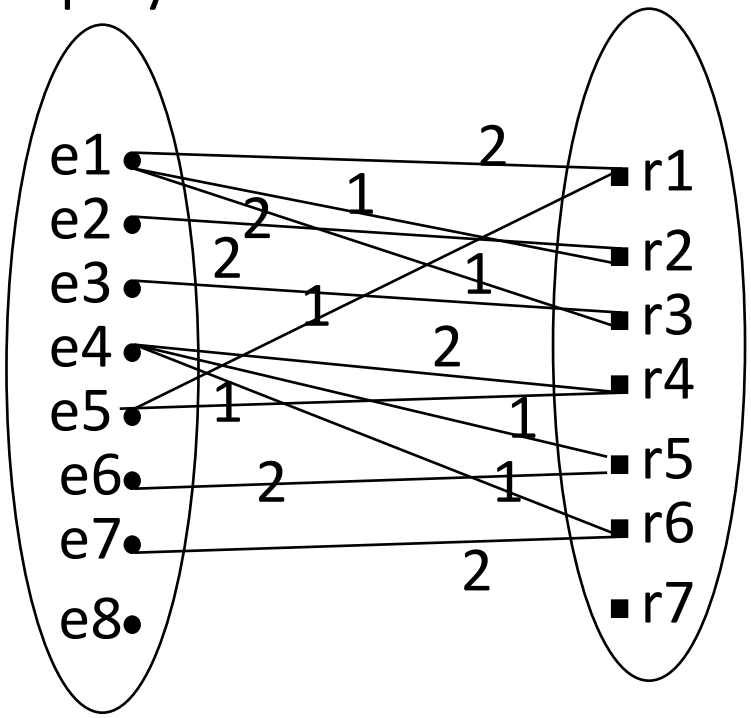
One to Many



Many to Many

# Relationship Types - Roles

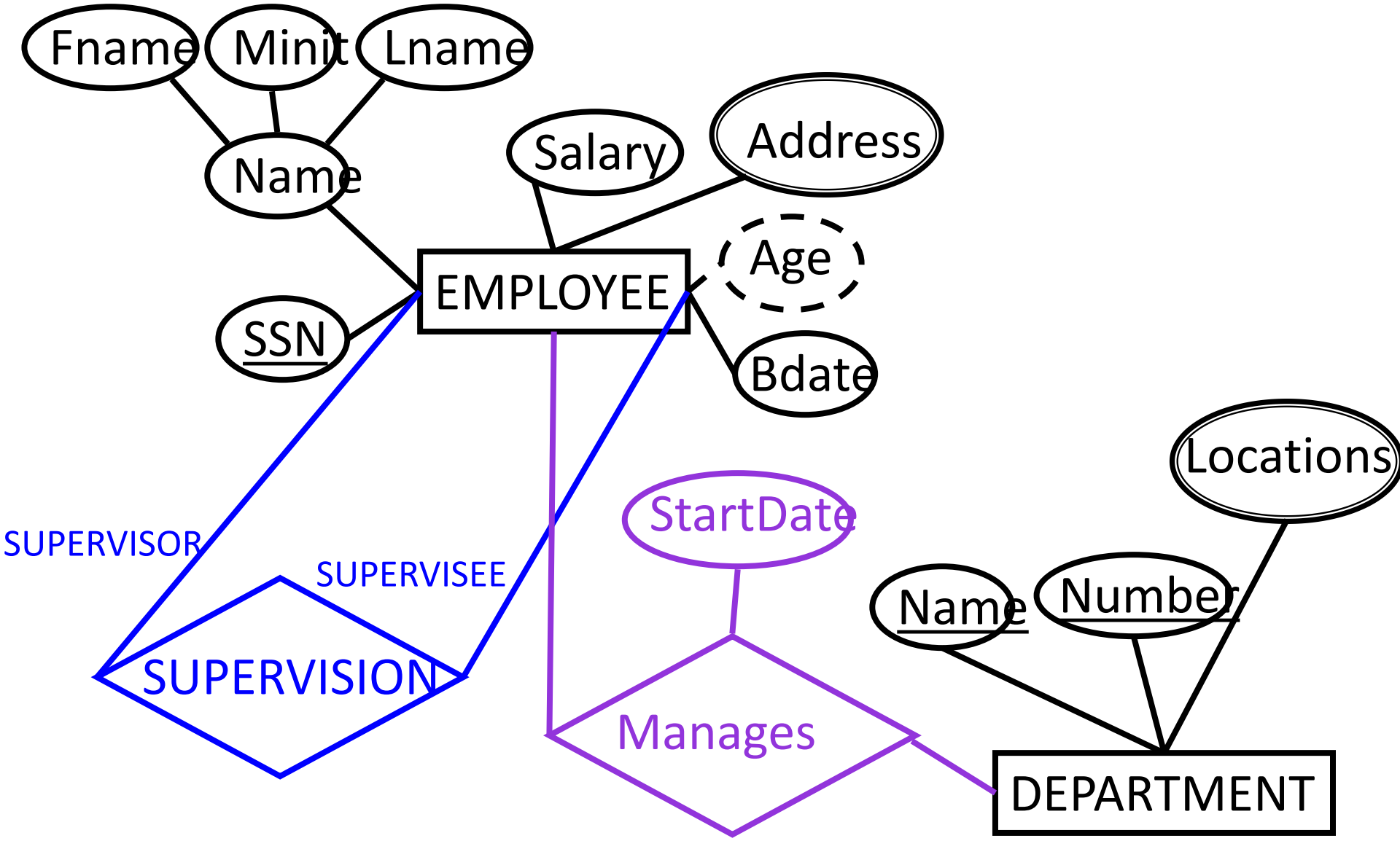
Roles of distinct entity types in a relationship are not necessary. But are required if the same entity type participates in the relationship. For example, a Supervision relationship type relates one Employee (in the role of Subordinate) to another Employee (in the role of supervisor). This is called a recursive relationship type.




1 - Supervisor Role

2- Subordinate Role

# Visualizing of Relationship Types



# Structural Constraints

Cardinality ratio (of a binary relationship)

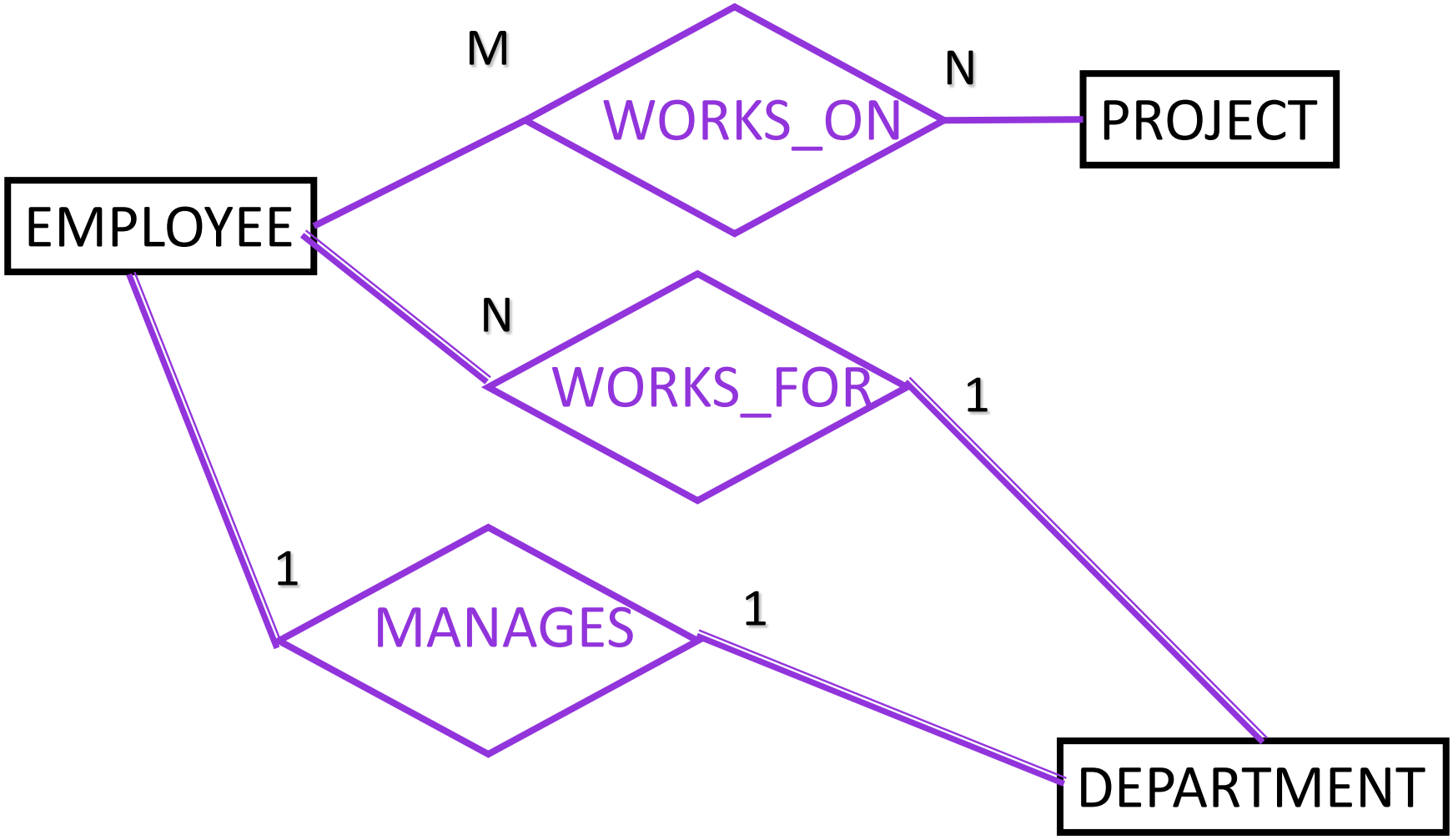
- 1:1 for binary relationship type E1:E2, each entity of entity type E1 can be related to an entity of entity type E2, and each entity of entity type E2 can be related to an entity of entity type E1. For example, MANAGES between EMPLOYEE and DEPARTMENT.
- 1:N for binary relationship type E1:E2, each entity of entity type E1 can be related to numerous entities of entity type E2, and each entity of entity type E2 can be related to an entity of entity type E1. For example, WORKS_FOR between DEPARTMENT and EMPLOYEE.
- M:N for binary relationship type E1:E2, each entity of entity type E1 can be related to numerous entities of entity type E2, and each entity of entity type E2 can be related numerous entities of entity type E1. For example, WORKS_ON between PROJECT and EMPLOYEE.

# Structural Constraints

Participation constraint whether the existence of an entity depends on its being related to another entity via the relationship type.

- Total participation (existence dependency) - every entity in "the total set" of entities must be related to other entities via a relationship. For example, Employee WORKS_FOR Department.
- Partial Participation - some or "part of the set of"entities are related to other entities via a relationship. For example, Employee MANAGES Department.

# Visualizing of Structural Constraints



# Notation for Structural Constraints

For 1:1 Relationship type R, place 1 on either side of the relationship type R.

For 1:N Relationship type between E1 and E2 (i.e., E1:E2), place N on the edge between E1 and R, and 1 between E2 and R.

For M:N Relationship type R, place M and N on either side of relationship type R.

For total participation of E1 in a relationship type R, make the edge between E1 and R double line.

# Alternate Notation for Structural Constraints

Alternate (min, max) Notation:

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least min and at most max relationship instances in R.
- Default (no constraint): min = 0, max = n.
- Must have min <= max, min >= 0, max >=1.
- Derived from mini-world constraints

# Structural Constraints Examples

- A department has exactly one manager and an employee can manage at most one department.
  - Specify (1,1) for participation of Department in Manages
  - Specify (0,1) for participation of Employee in Manages
- An employee must work in exactly one department but a department can have any number of employees
  - Specify (1,1) for participation of Employee in Works_For
  - Specify (0,n) for participation of Department in Works_For

# Relationship Types as Attributes

In some cases one can represent a relationship type as an attribute of an entity type

- A 1:1 relationship type can be replaced by an single valued attribute in any of the participating entity types.
- A 1:n relationship type can be replaced by an single valued attribute in the "n" side entity type, but require multivalued attribute in the "1" side entity type.
- But for m:n relationship type we have to use multivalued attribute in both the entity types (for a binary relationship type).

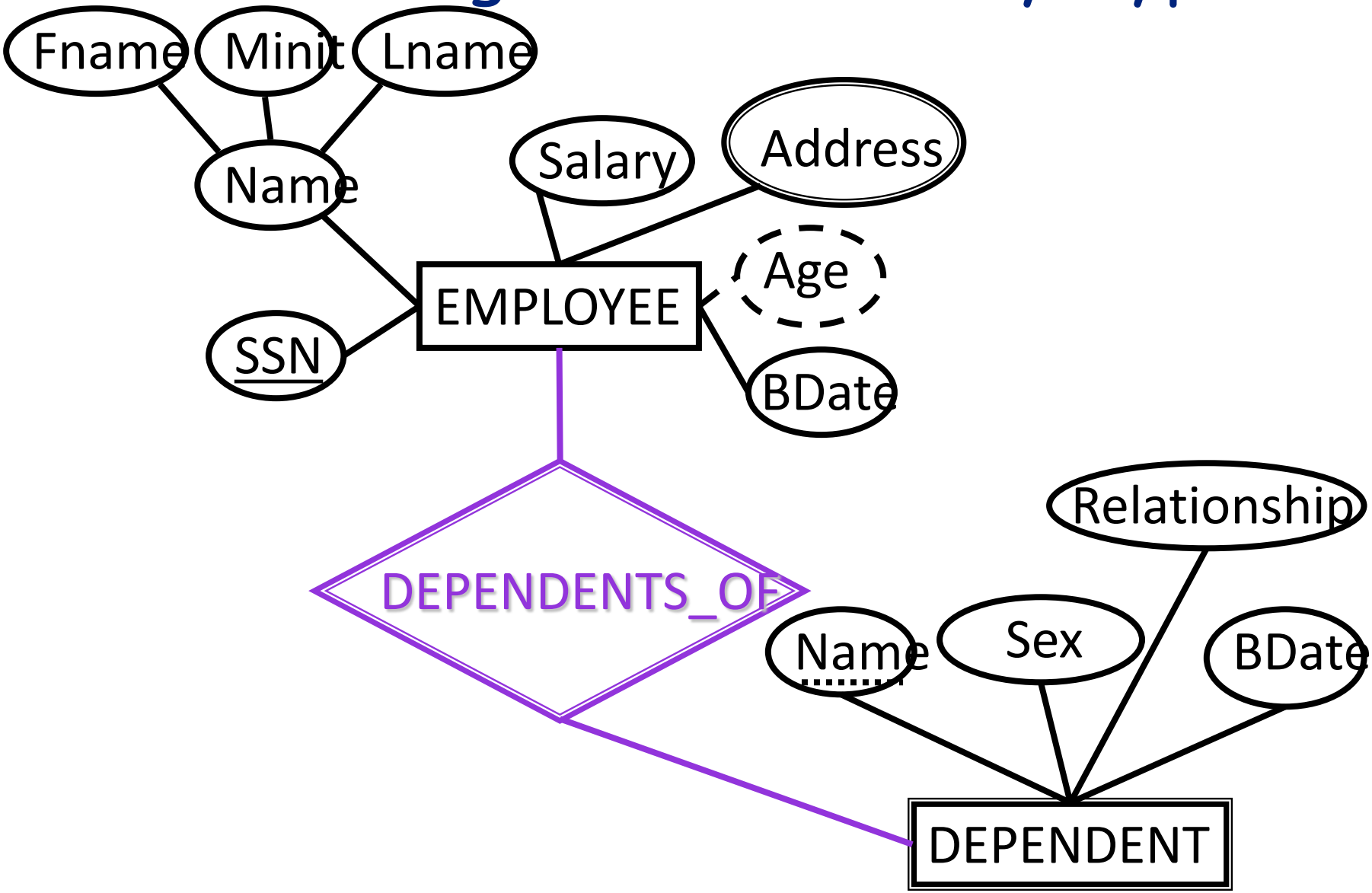
Relationships are represented by attributes in functional and object-oriented data models.

# Weak Entity Types

- An entity type that does not have a key attribute
- A weak entity type must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

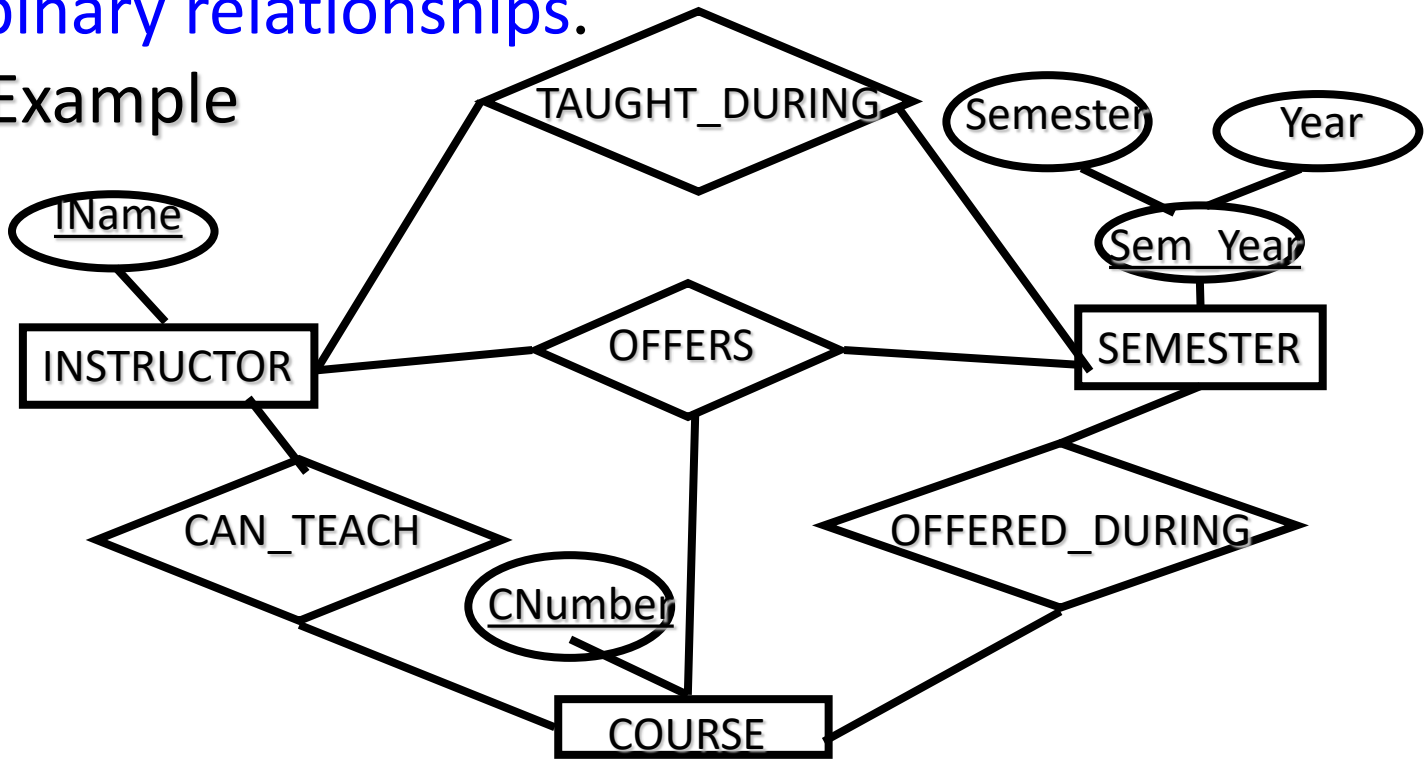
Example: Suppose that a Dependent entity is identified by the dependent's first name and birthdate, and the specific Employee that the dependent is related to. Dependent is a weak entity type with Employee as its identifying entity type via the identifying relationship type DEPENDENTS_OF.

# Visualizing of Weak Entity Types



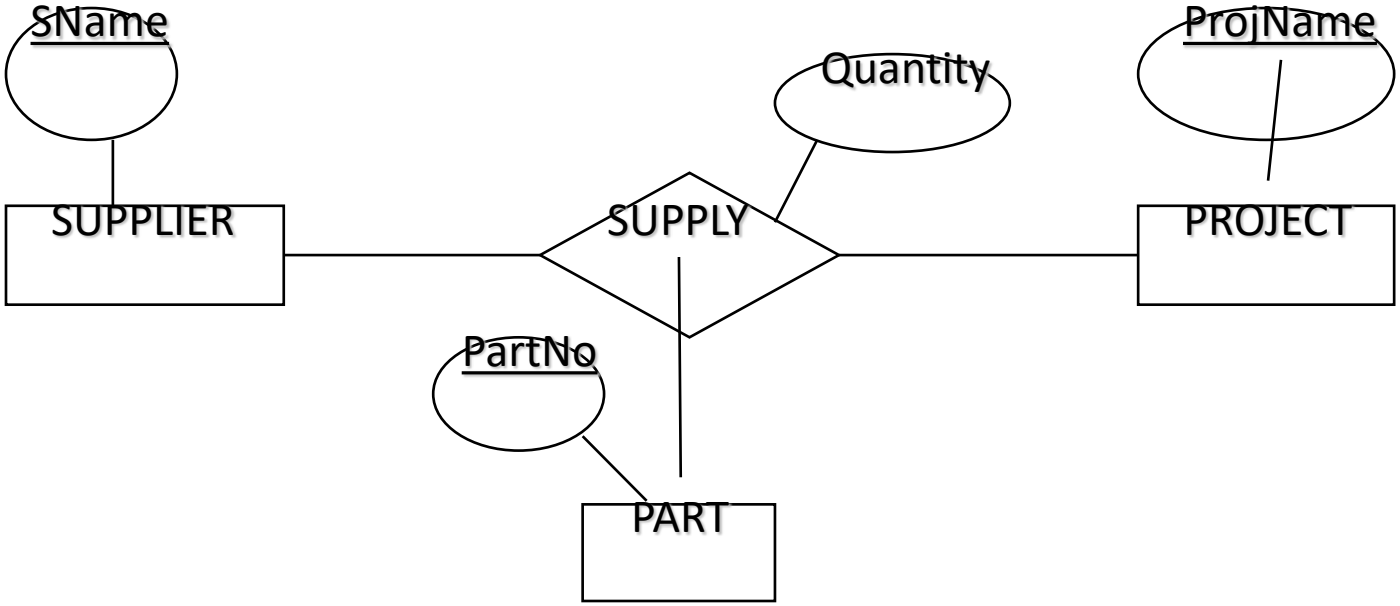
# Relationships with higher degree

- Relationship types of degree 2 are called binary
- Relationship types of degree three are called ternary and of degree n are called n-ary
- In general, an n-ary relationship is not equivalent to n binary relationships.
- Example



# Relationships with higher Degree

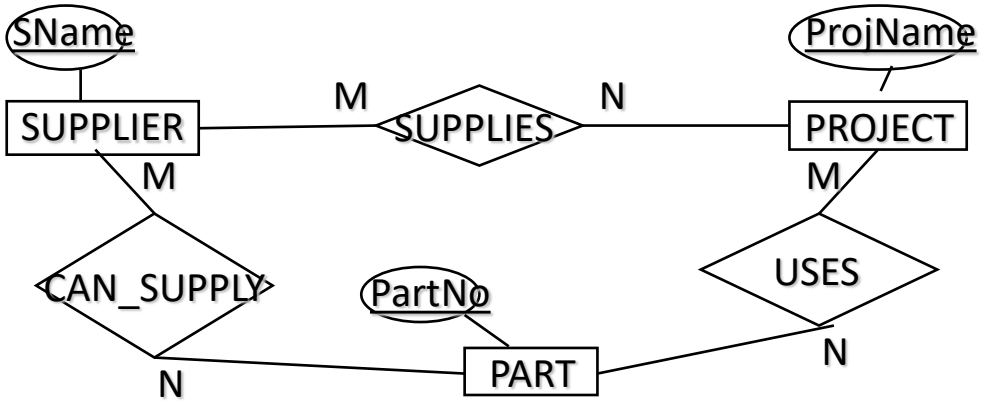
(a)




Above ternary relationship models the fact that a supplier supplies a part (attribute quantity denoting number of parts) to a project .

# Relationships with higher Degree

(b)

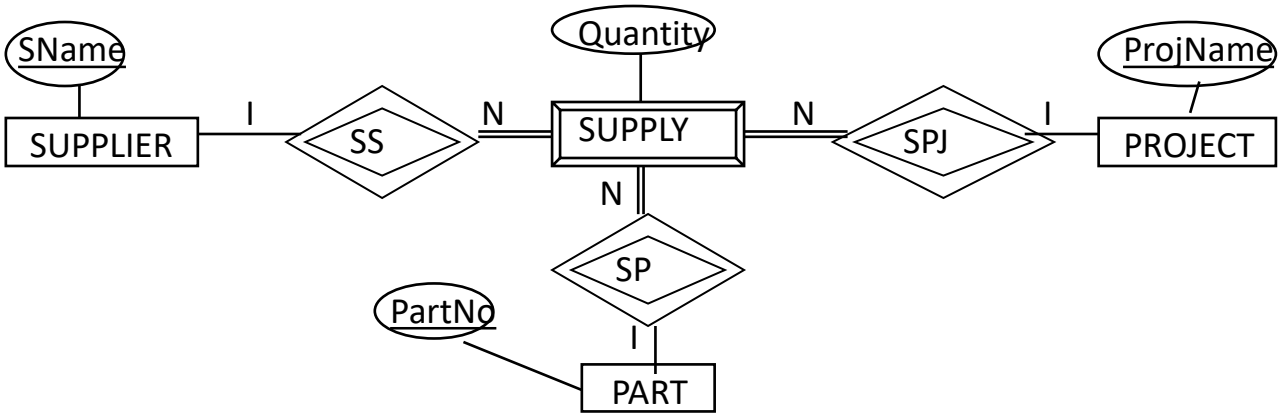



The above ER model shows three binary relationship types, namely, Supplies between Supplier and Project, Uses between Part and Project, and Can_Supply between Part and Supplier.

This is not the same as the ternary relationship type Supply shown in the previous page.

# Relationships with higher Degree

(c)



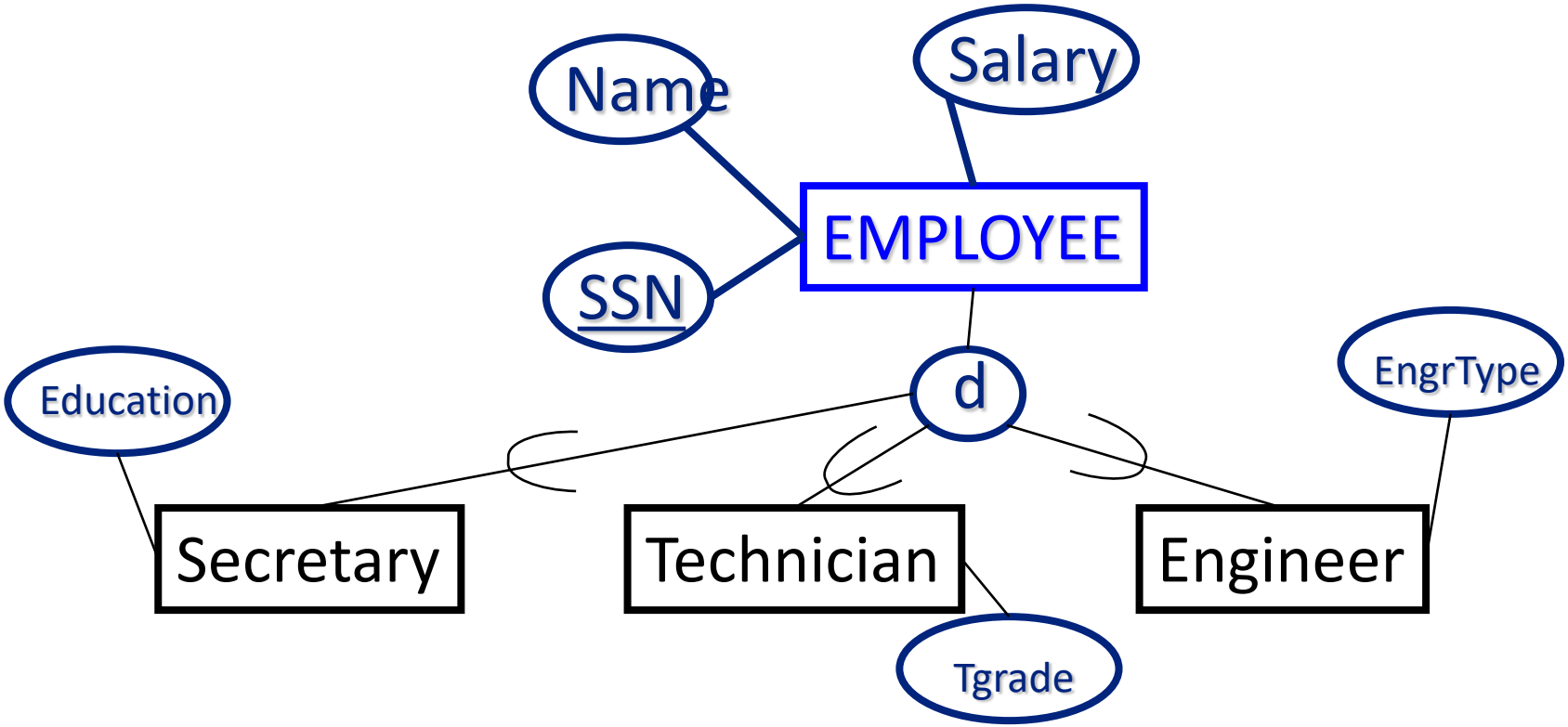

The above ER model shows how the original ternary relationship type can be modeled by using three binary relationship types. This is done by making original “Supply” relationship type a weak entity with three identifying entity types Supplier, Project and Part. And the corresponding identifying relationship types, SS, SPJ, and SP.

# Enhanced ER Model

- Specialization
- Generalization
- Aggregation

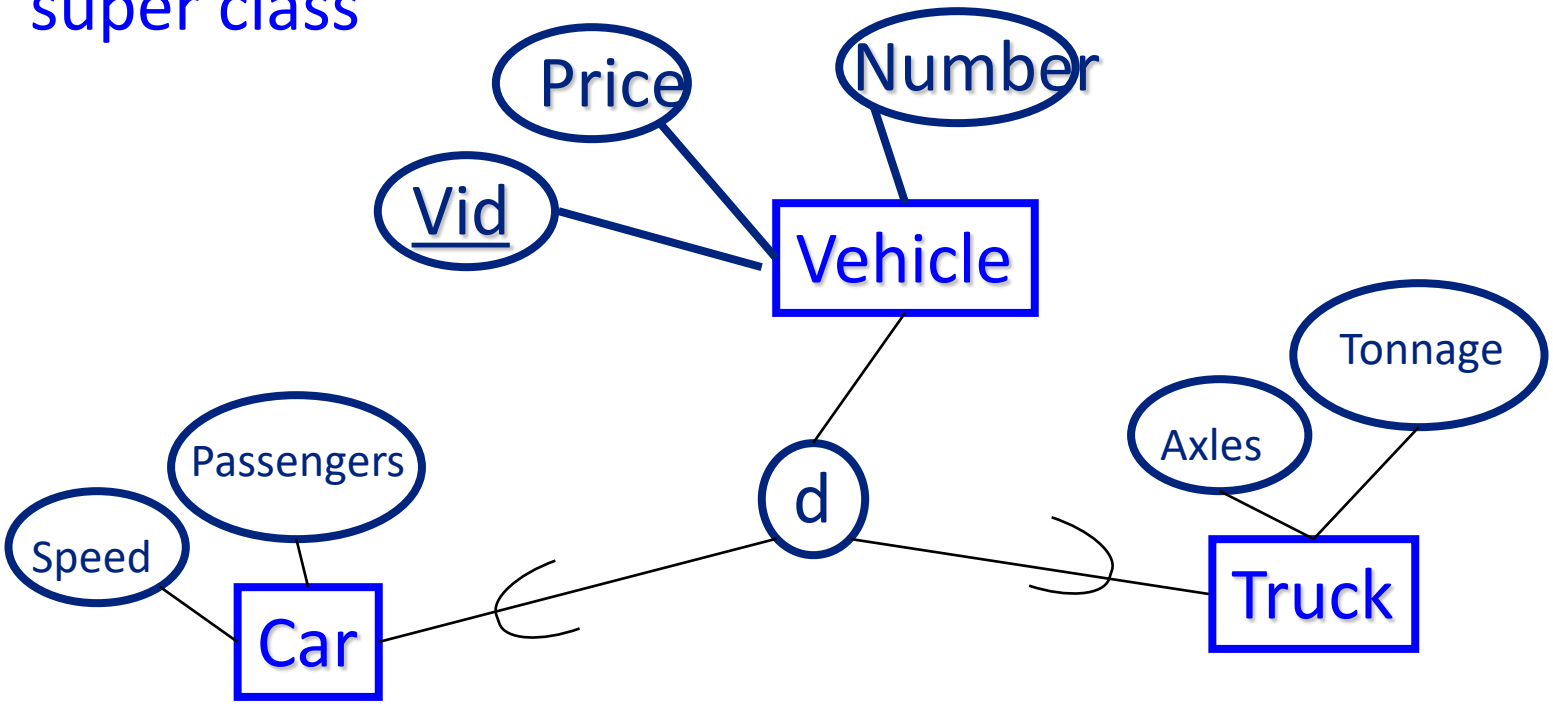
# Specialization

- Specialization is the process of defining a set of subclasses of an entity type (called the superclass of specialization) – support for “is-a” relationship
- For example set of subclasses {secretary, engineer, technician} of super class employee



# Generalization

- Suppress differences of different entity types, identify their common attributes, and generalize them into a super class



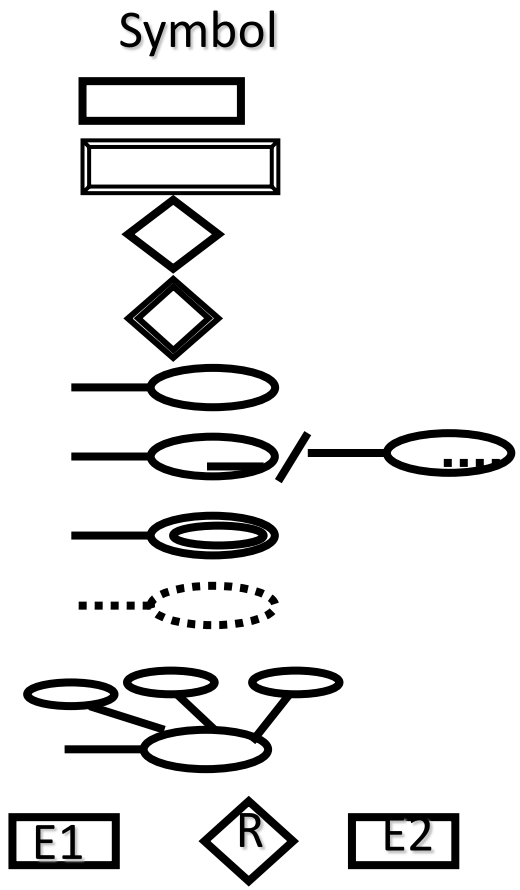
# Constraints on Specialization & Generalization

- Disjoint, Total
- Disjoint, Partial
- Overlapping, Total
- Overlapping, Partial

# Aggregation

- Aggregation is an abstraction concept for building composite objects from their component objects
- Support for “is part of” relationship

# ER-diagram Notation




Notation

Entity

Weak Entity

Relationship Types

Identifying Relationship Type

Attribute

Key Attribute / Partial Key

Mutivalued Attribute

Derived Attribute

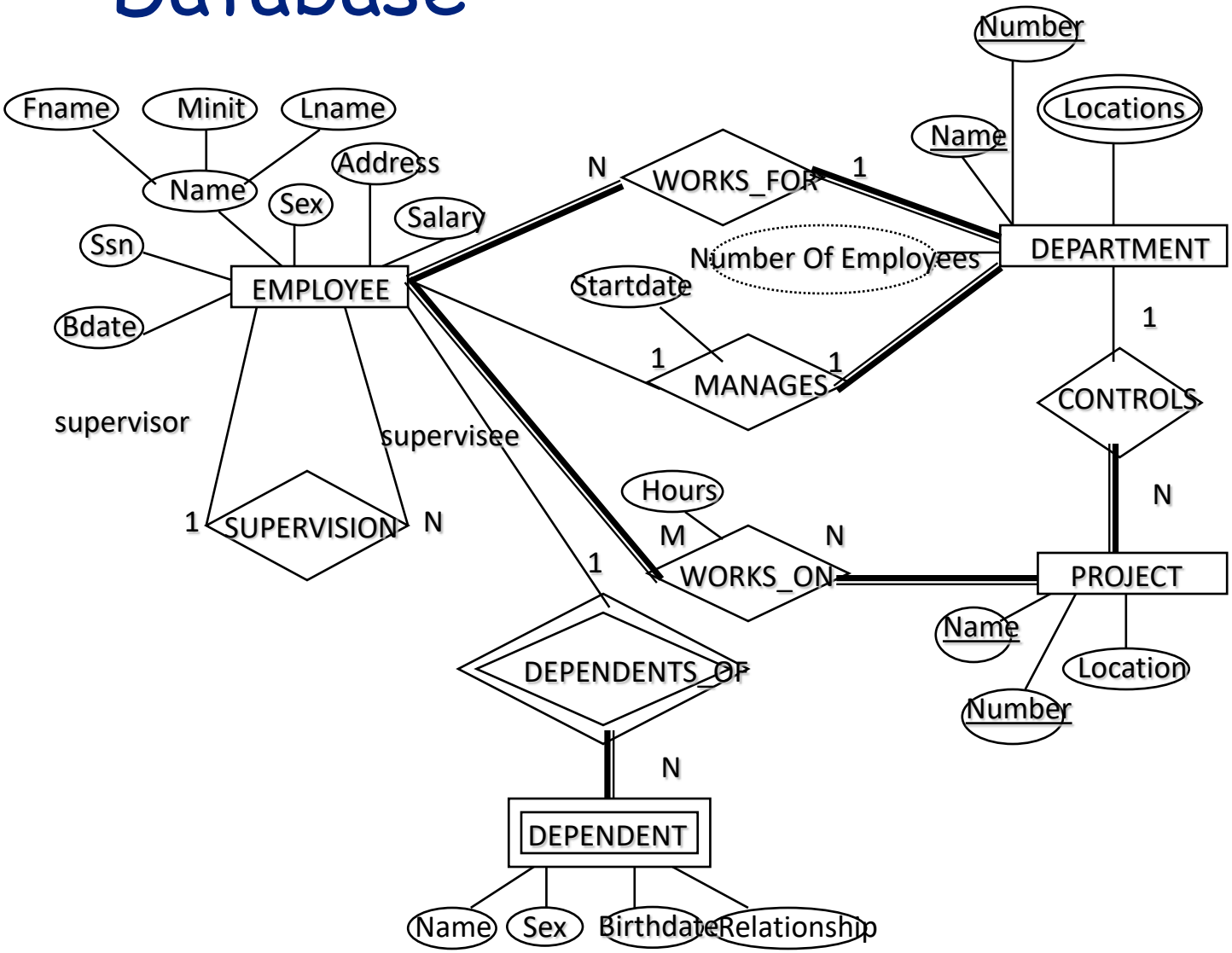
Composite Attribute

Total Participation of E2 in R

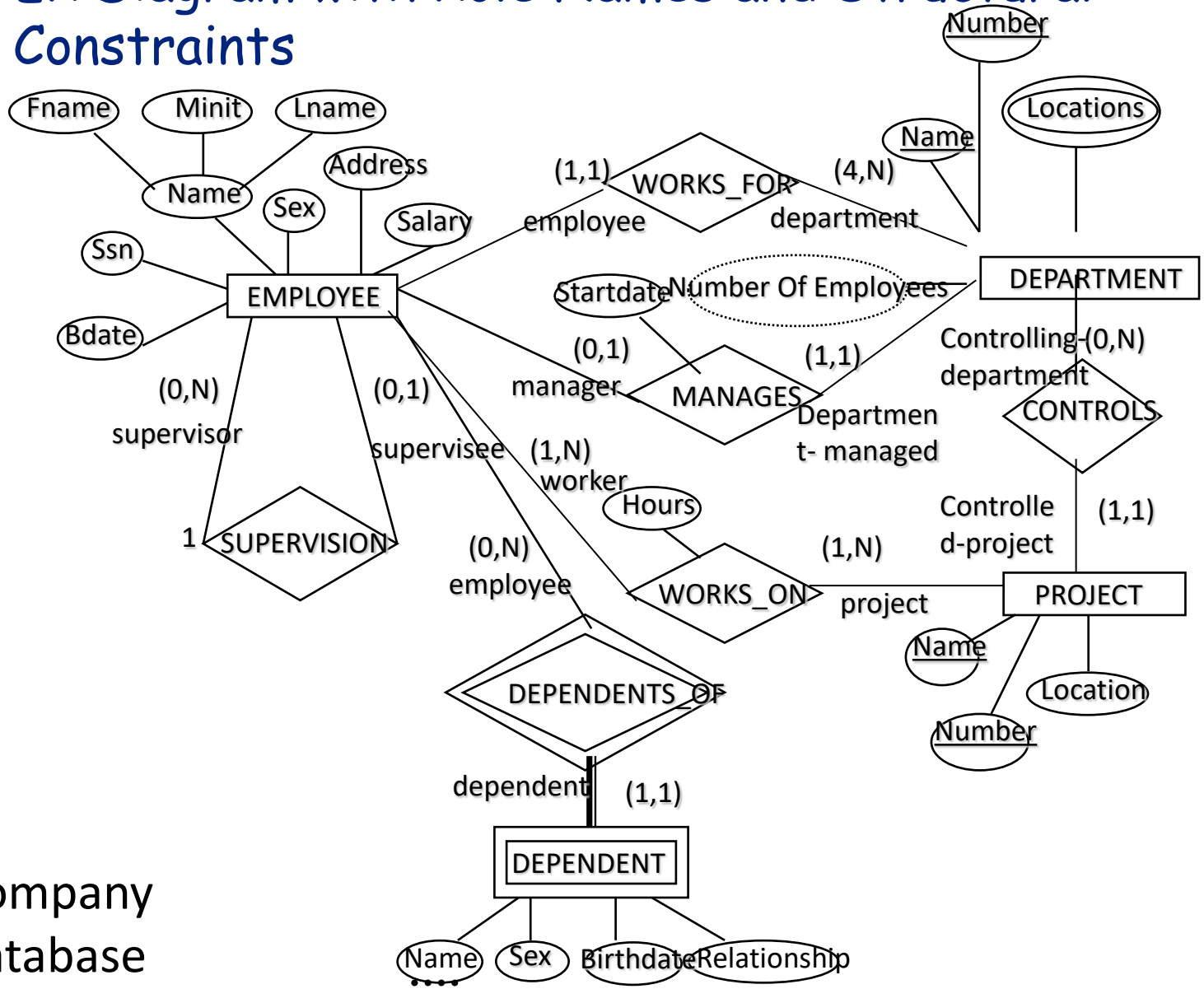
Cardinality Ratio 1 : N for E1 : E2 in R

Structural Constraint (min, max) of Participation of E in R

# ER Diagram for Company Database



# ER Diagram with Role Names and Structural Constraints




Company database

# Data Abstraction & Knowledge Representation

- Apply abstraction process to identify common properties – while disregarding insignificant differences
- Provide concepts, constraints, operations, and languages for defining data and representing knowledge
- KR broader than data models
- KR includes reasoning mechanisms

# Abstraction Concepts in SDMs & KR

- Classification and Instantiation
  - Define object classes – 'IS-AN-INSTANCE-OF'
- Identification
  - Classes and objects are made uniquely identifiable
  - To distinguish among database objects and classes and relate database objects to their real-world counterparts

# Abstraction Concepts in SDMs & KR

- Specialization & Generalization
- Aggregation & Association
  - Aggregation builds composite/complex objects – models 'IS-PART-OF'
  - Association relates objects from independent classes – models 'IS-ASSOCIATED-WITH'

# Ontology & Semantic Web

- Ontology
  - Is similar to a conceptual schema, but with more knowledge, rules, and exceptions
  - 'Specification of Conceptualization' – includes both specification & conceptualization
- Semantic Web
  - Attempts to create knowledge representation models to allow meaningful information exchange and search among machines

# Examples of Ontology

- Thesaurus – dictionary – glossary: describes relationships between words that represent various concepts
- Taxonomy: how concepts of particular area of knowledge are related using structure similar to specialization & generalization
- Database Schema
- Logic theory: propositional logic – predicate calculus, frames, etc.