Chapter 3

Data Modeling Using the Entity-Relationship (ER) Model (from E&N and my editing)

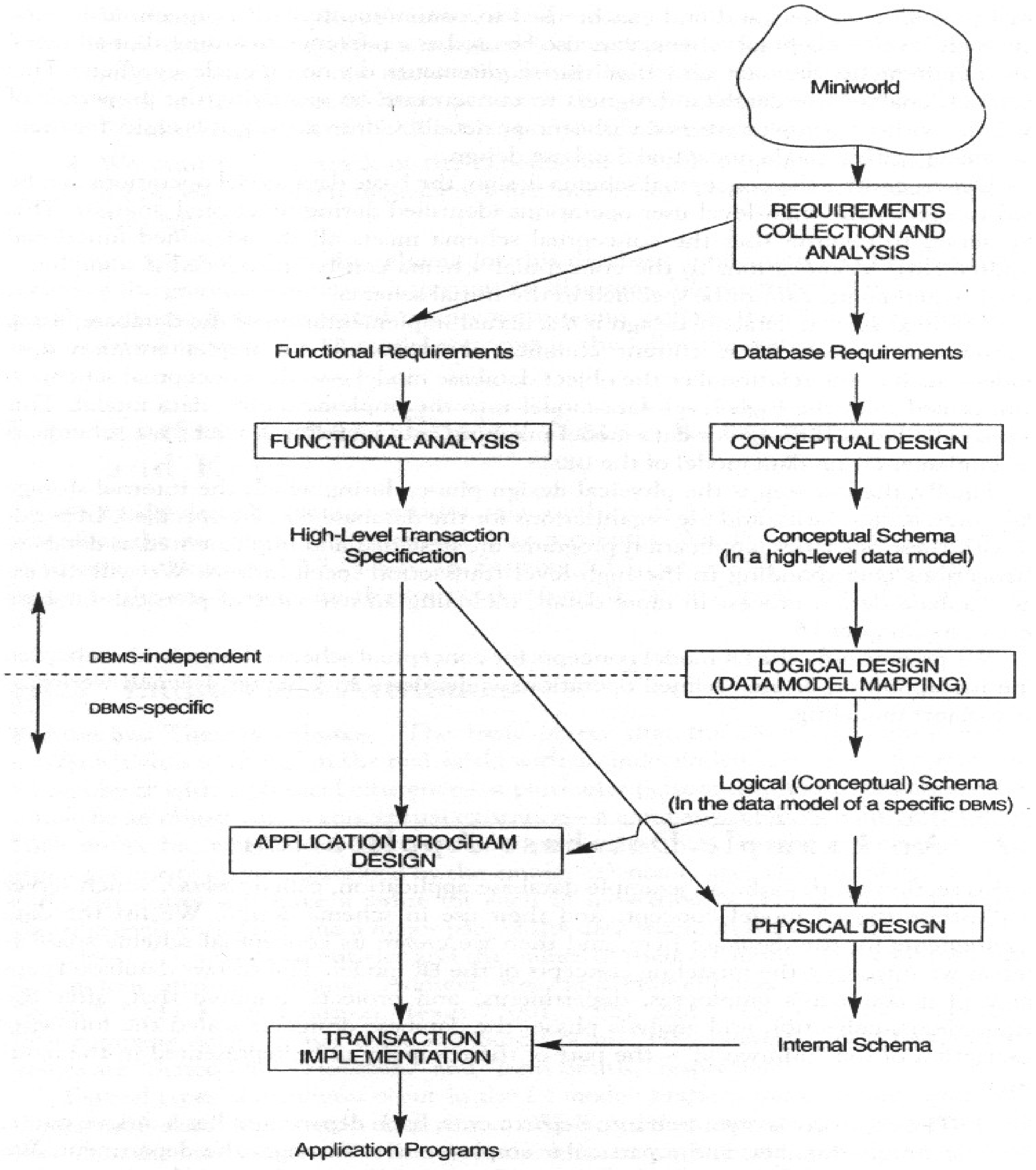
Prof.Peter Chen

- The founder ER Data Model
- PhD: Harvard

Chapter Outline

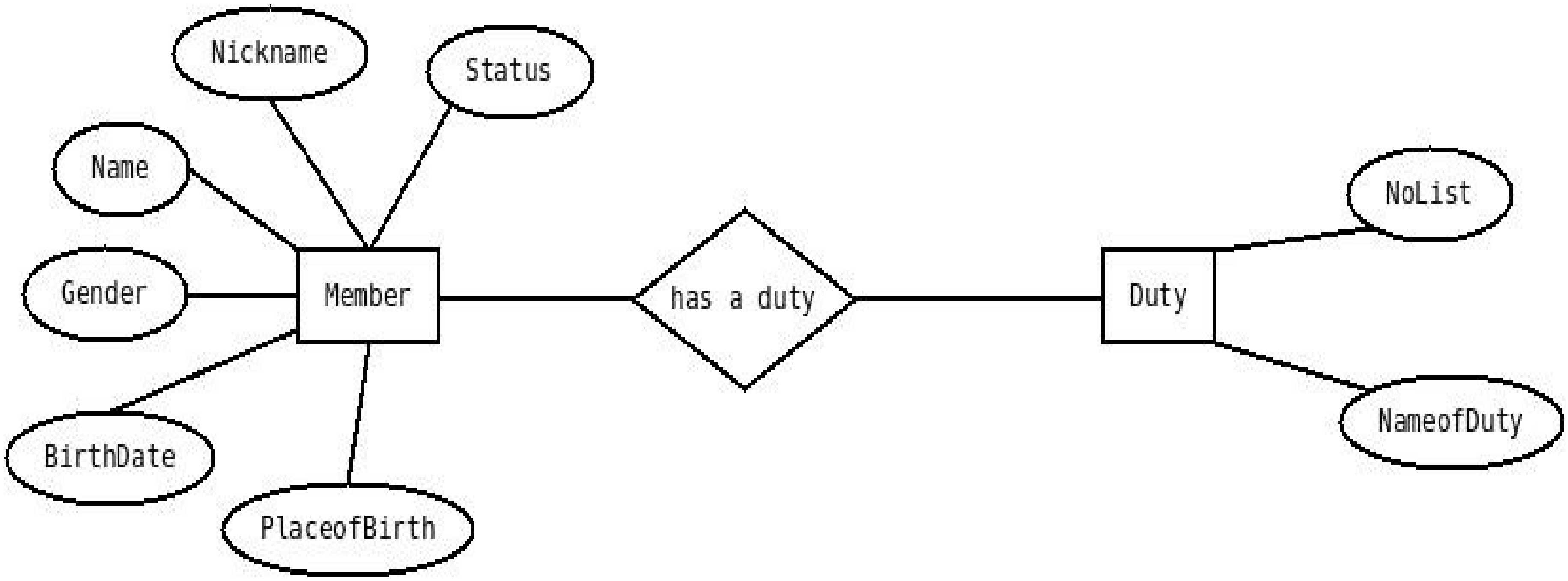
- Phase of Database Design
- Example Database Application (COMPANY)
- ER Model Concepts
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram for COMPANY Schema

- Phase


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Database Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

The simple one

- Requirement of family database
  - In a family there are a member of family
  - Each member has data including: status, name, place of birth, birthdate, gender or maybe also nickname
  - There is a list of home duty ex: wash dishes, clean living room or dining room etc


Nickname
Status
Name
Gender
Member
BirthDate
PlaceofBirth
has a duty
Duty
NoList
NameofDuty

Example COMPANY Database

- Requirements of the Company (oversimplified for illustrative purposes)
  - 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 of the department manager.
  - Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.

Example COMPANY Database (Cont.)

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
- 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 track of their name, sex, birthdate, and relationship to employee.

ER Model Concepts

- 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 ProductX PROJECT
- Attributes
  - Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘
  - Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, subrange, enumerated type, …

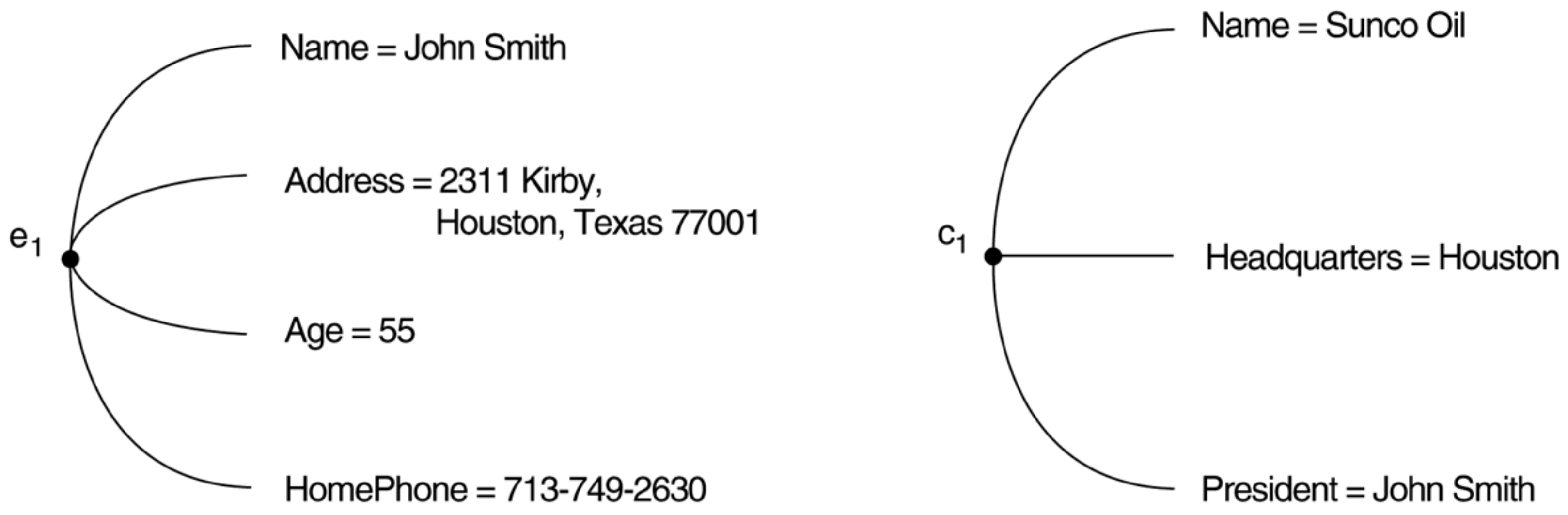
Types of Attributes (1)

- Simple
  - Each entity has a single atomic value for the attribute.
  - For example, SSN or Sex.
- Composite
  - The attribute may be composed of several components.
  - For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite.
- Multi-valued
  - An entity may have multiple values for that attribute.
  - For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

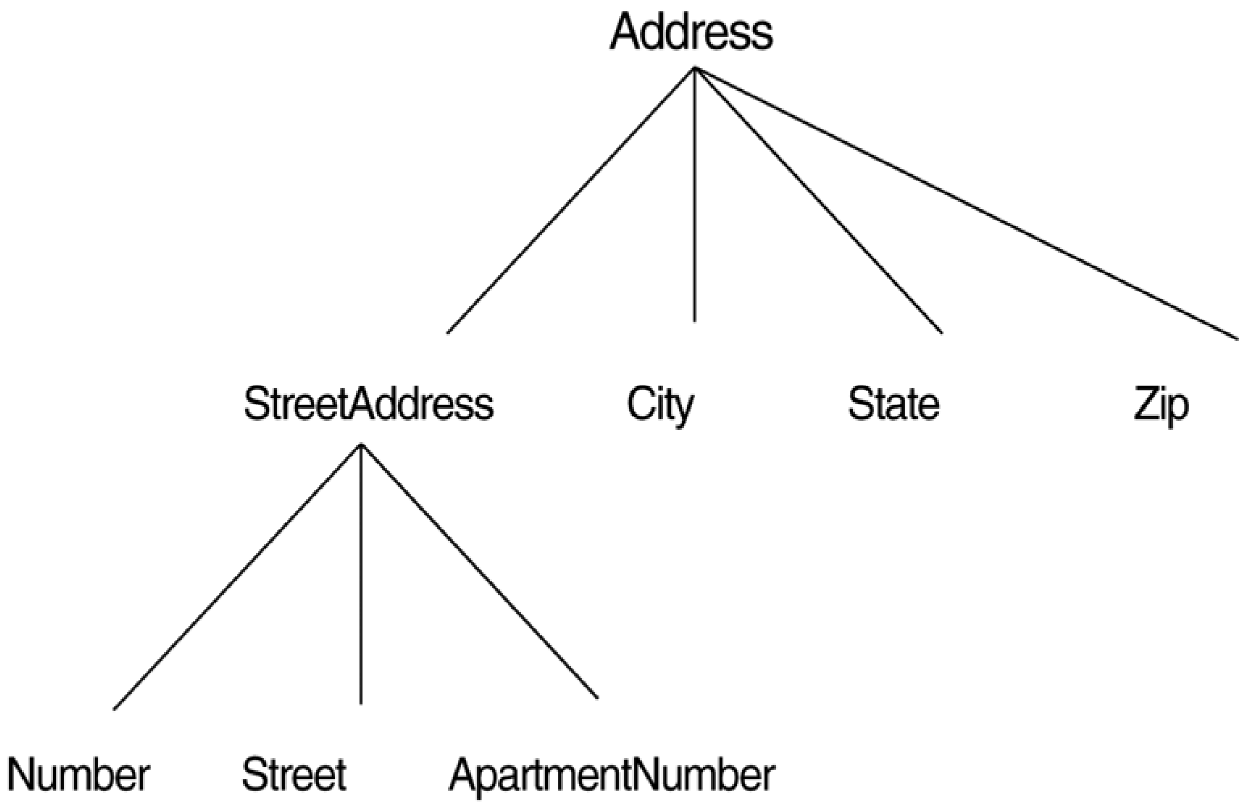
Types of Attributes (2)

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.
  - For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}.
- Stored vs Derived Attibutes
  - Stored – regular attribute
  - Derived -- attribute which is calculated from a stored attribute
  - BirthDate vs Age
- Null Values
  - “nothing”, not zero, not blank space!
  - Ex. College Degree for Employee entity

Entity with attrb values



Hierarchy of attrb



Complex Attrb

- Composite and multi-valued attributes can be nested in an arbitrary way
- () for nesting
- {} for multi value
- Example

{Address&Phone (

{Phone(AreaCode, PhoneNumber)},

Address(StreetAddress(Number, Street, ApartmentNumber),

City, State, Zip)

)}

Entity Type

- It defines the set of possible entities with the same attributes
- For example, ‘employee type’ is the set of employees in the company

Key Attrb of Entity

- What makes an entity unique?
- An employee: SSN
- A company: name
- A project: number, name
- A purchase slip may have 2 keys: date & time

Key Attrb

- Selection of the keys is an important part of the database design
  - It affects integrity validation and performance
  - Declaring an attribute as a key, and declaring it 'duplicates not allowed' we can prevent users form entering erroneous duplicate data
  - The key can also maintain integrity by linking the key with a key in another table.

- A key attribute may be composite.
  - For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
- For example, the CAR entity type may have two keys:
  - VehicleIdentificationNumber (popularly called VIN) and
  - VehicleTagNumber (Number, State), also known as license_plate number.

ENTITY SET corresponding to the ENTITY TYPE CAR

CAR

Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, (Color)

car_1

((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1999, (red, black))

car_2

((ABC 123, NEW YORK), WP9872, Nissan 300ZX, 2-door, 2002, (blue))

car_3

((VSY 720, TEXAS), TD729, Buick LeSabre, 4-door, 2003, (white, blue))

.

.

.

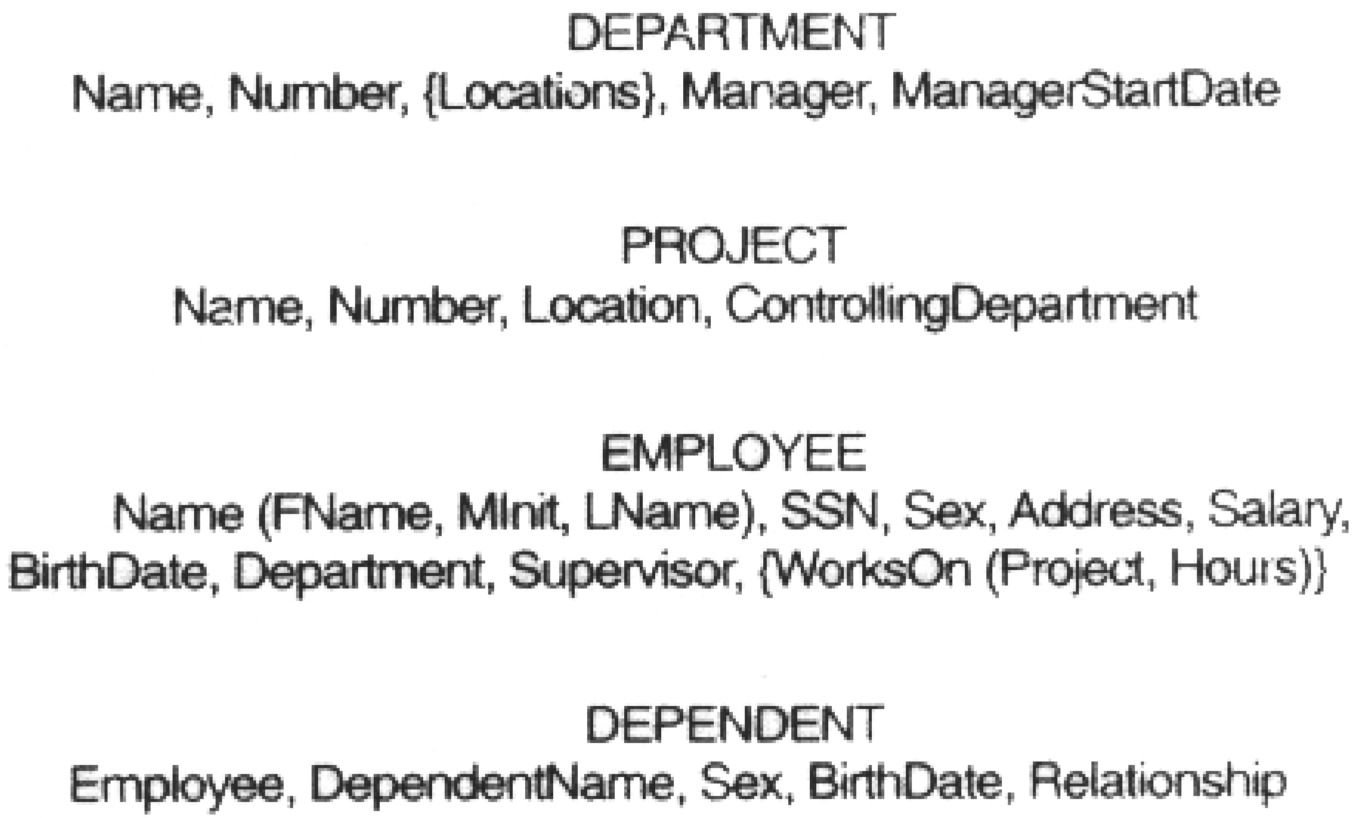
Selection Key Attrb

- The key should be an attribute that doesn't change
  - Example: ssn, employee_ID, SKU (stock-keeping-unit), license plate number.
- The key can't be a null. It must have a valid value
  - Example: Actual graduation date for a student would be a bad choice
- Avoid using keys that have intelligence or codes built in
  - Example: A building code (which might later be subject to change)

Value Set

- Possible values of an attribute
- Usually:
  - Numeric
  - Text
  - Boolean
  - Etc.
- But can be more specific, for example:
  - Birth date must be > 1850 but before <(now)
  - Sex must be either female or male

Initial Entity Concept of COMPANY case



DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary, BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

Specifying Entity

- Entity – a class of persons, places, objects, events, or concepts about which we need to capture and store data.
- Named by a singular noun
  - Persons: agency, contractor, customer, department, division, employee, instructor, student, supplier.
  - Places: sales region, building, room, branch office, campus.
  - Objects: book, machine, part, product, raw material, software license, software package, tool, vehicle model, vehicle.
  - Events: application, award, cancellation, class, flight, invoice, order, registration, renewal, requisition, reservation, sale, trip.
  - Concepts: account, block of time, bond, course, fund, qualification, stock.

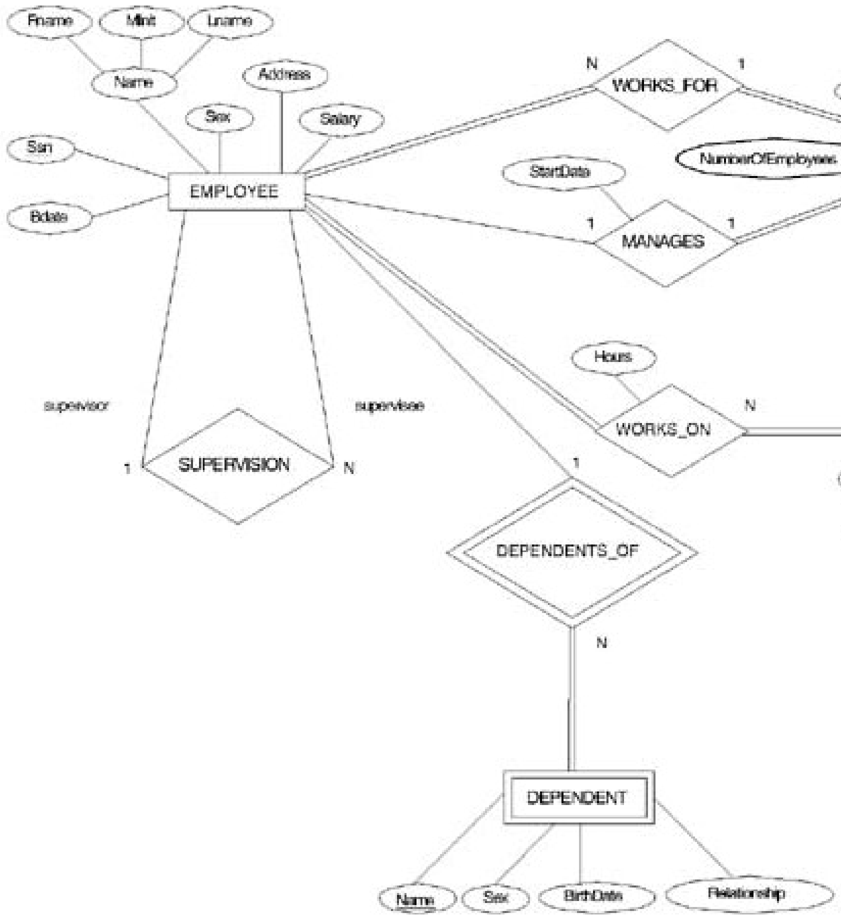
Weak Entity Types

- An entity that does not have a key attribute
- A weak entity 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

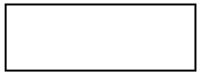
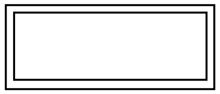
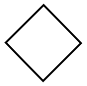
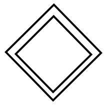
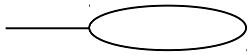
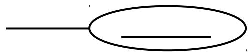
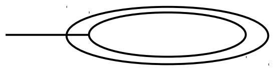
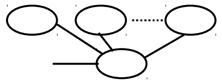
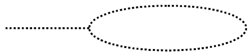
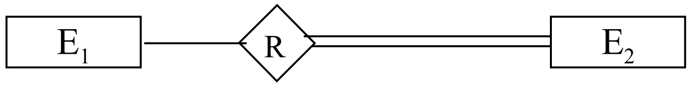
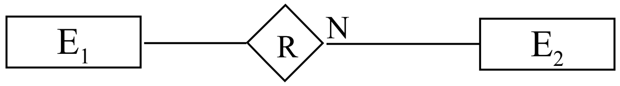
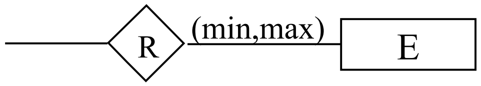
Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birhtdate, *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 DEPENDENT_OF

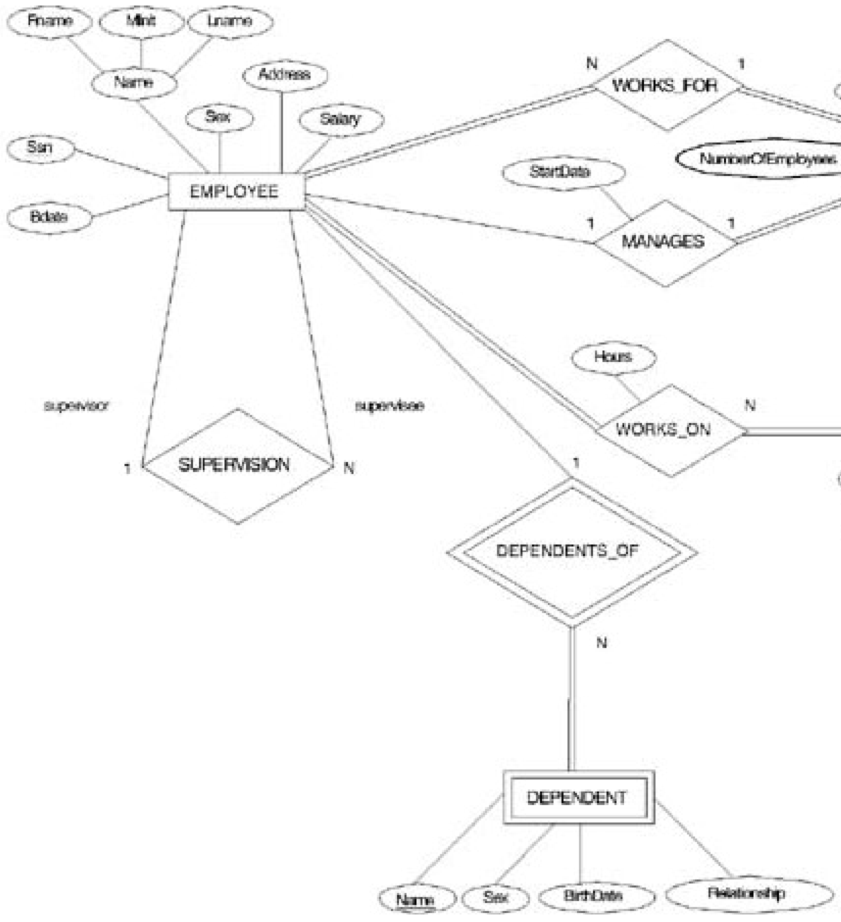
Weak Entity Type is: DEPENDENT Identifying Relationship is: DEPENDENTS_OF



SUMMARY OF ER-DIAGRAM NOTATION FOR ER SCHEMAS

Symbol	Meaning
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	IDENTIFYING RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E_2 IN R
	CARDINALITY RATIO 1:N FOR E_1:E_2 IN R
	STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

ER DIAGRAM – Entity Types are:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

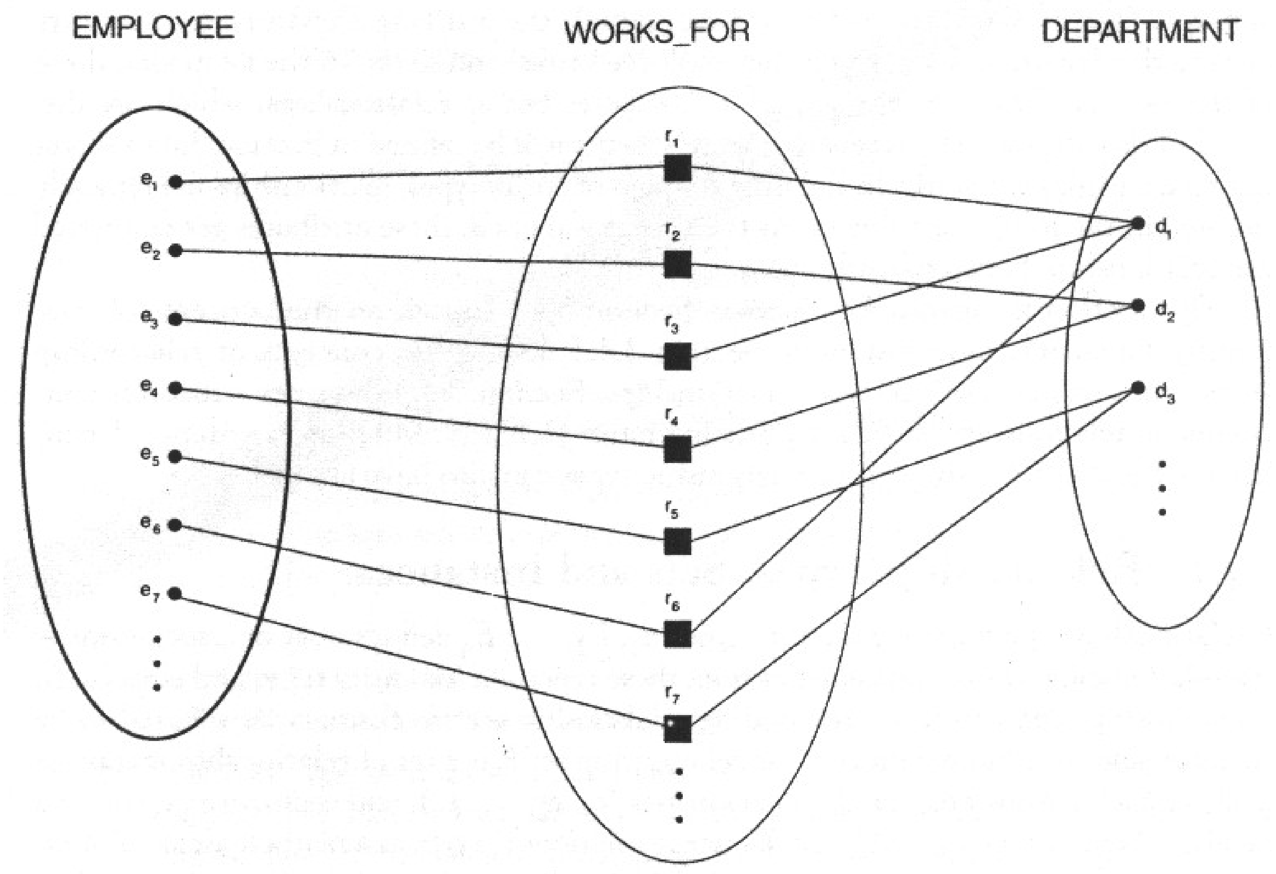


Relationships and Relationship Types

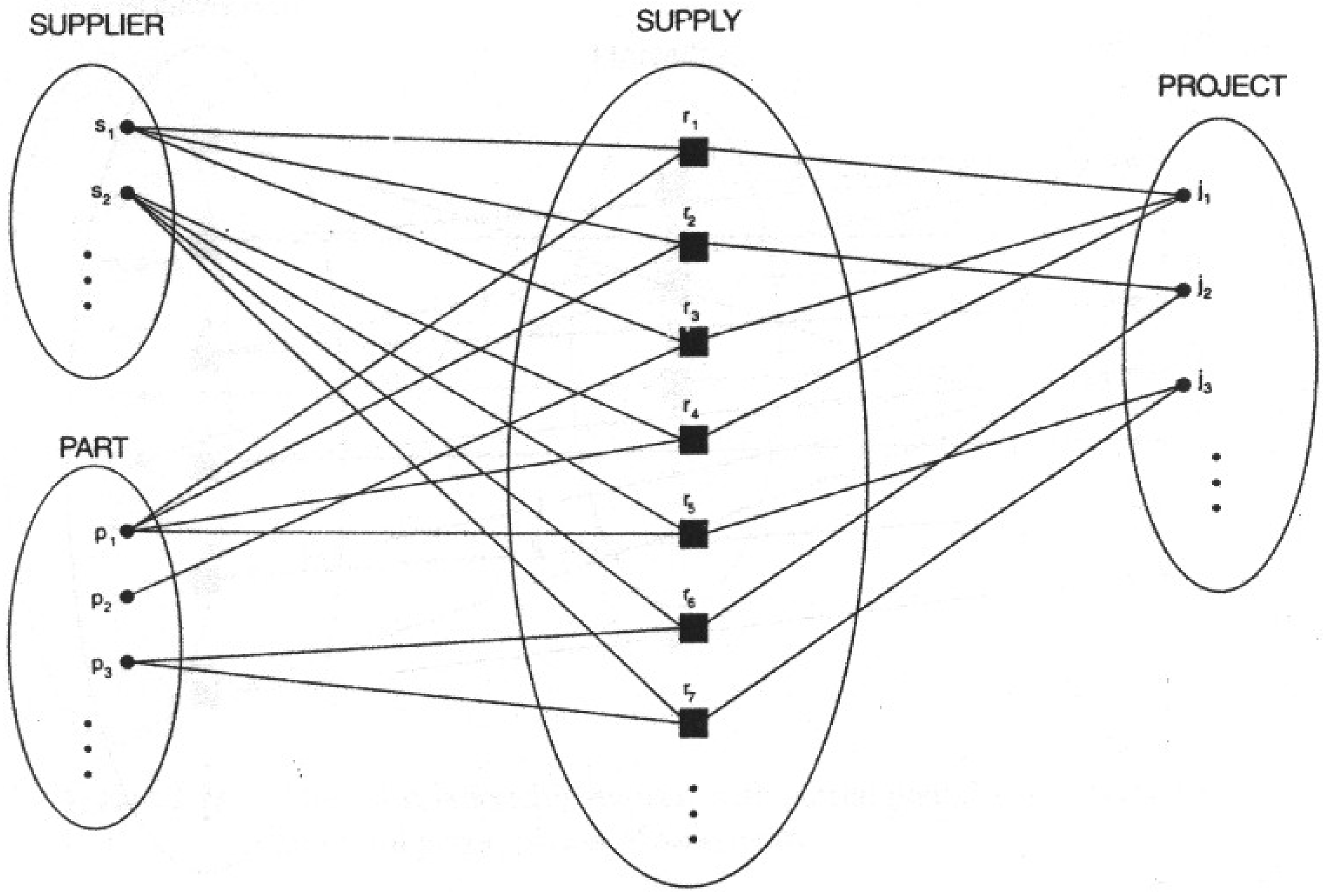
- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or
  - EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or
  - the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.

- Degree of a Relationship Type is the Number of Participating Entity Types
  - Binary Relationship → Between two entities
  - Ternary Relationship → Among three entities
  - N-ary Relationship → Among N entities
- More Than One Relationship Type Can Exist With the Same Participating Entity Types
  - MANAGES and WORKS_FOR are Distinct Relationships Between EMPLOYEE and DEPARTMENT Entity Types, but with different meanings and different relationship instances.
- Relationships are Directional
  - SUPPLIES: SUPPLIER to PARTS
  - SUPPLIERS: PARTS to SUPPLIER

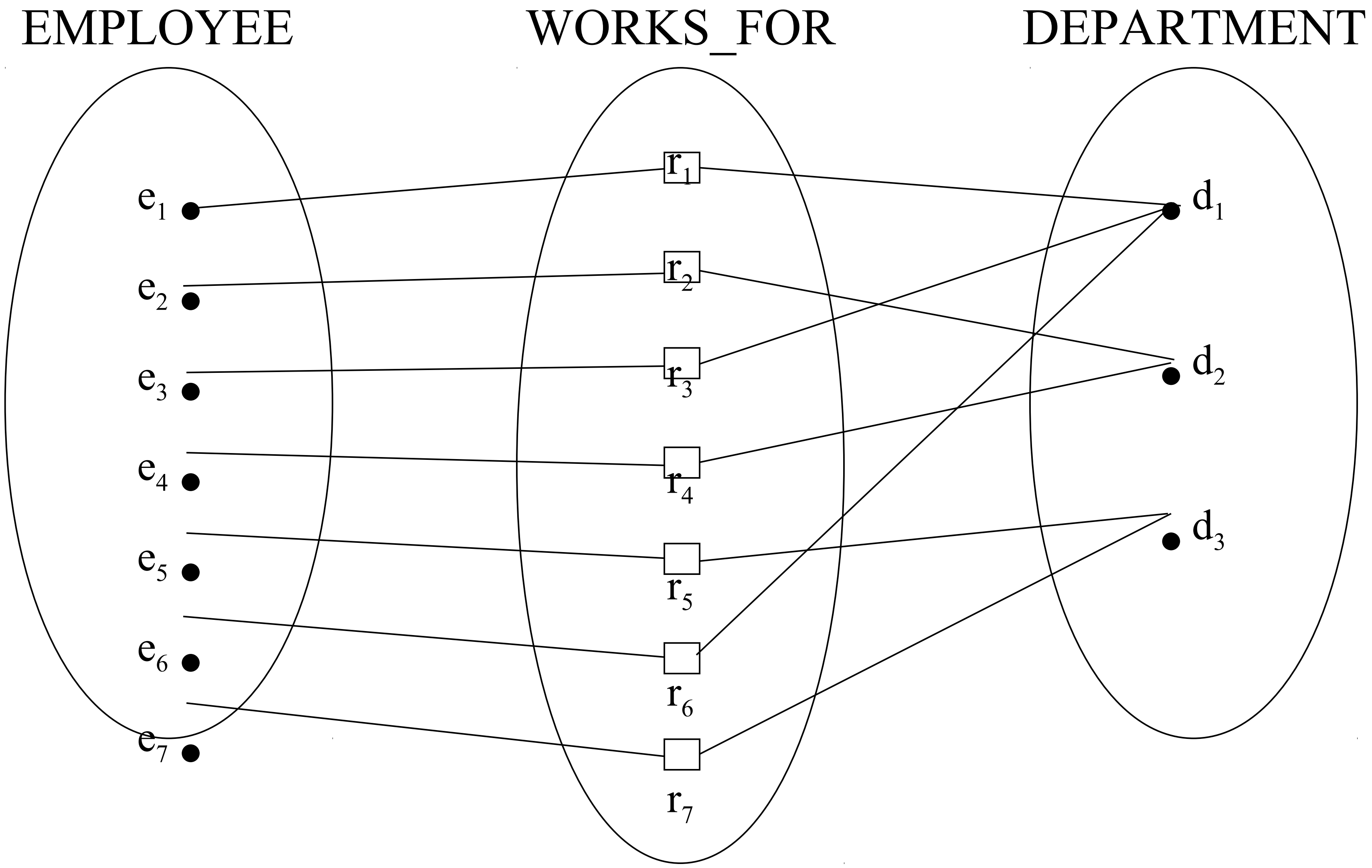
Binary Rel


EMPLOYEE
WORKS_FOR
DEPARTMENT
e_1
e_2
e_3
e_4
e_5
e_6
e_7
r_1
r_2
r_3
r_4
r_5
r_6
r_7
d_1
d_2
d_3

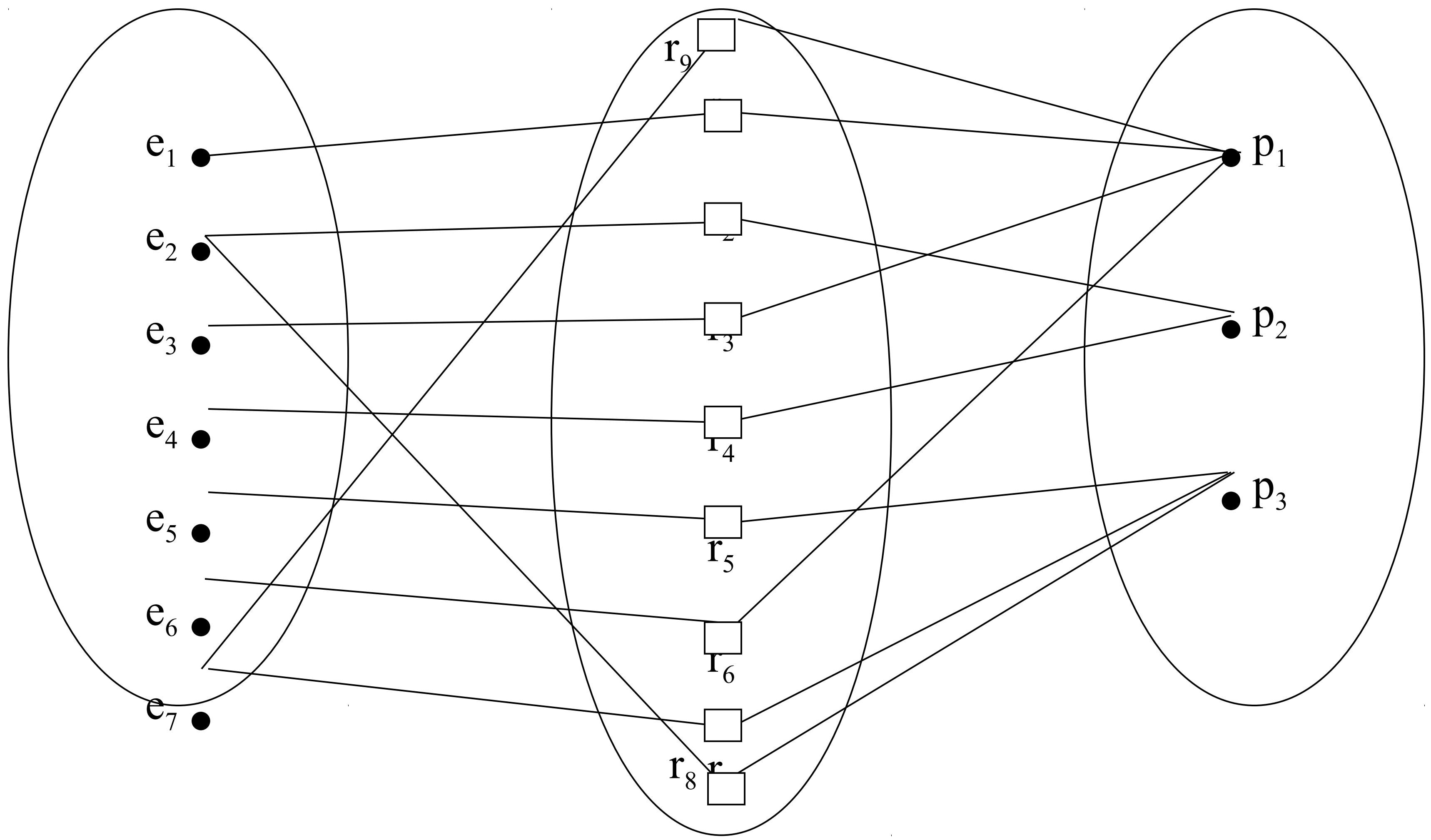
Ternary Rel


SUPPLIER
s_1
s_2
PART
p_1
p_2
p_3
SUPPLY
r_1
r_2
r_3
r_4
r_5
r_6
r_7
PROJECT
j_1
j_2
j_3

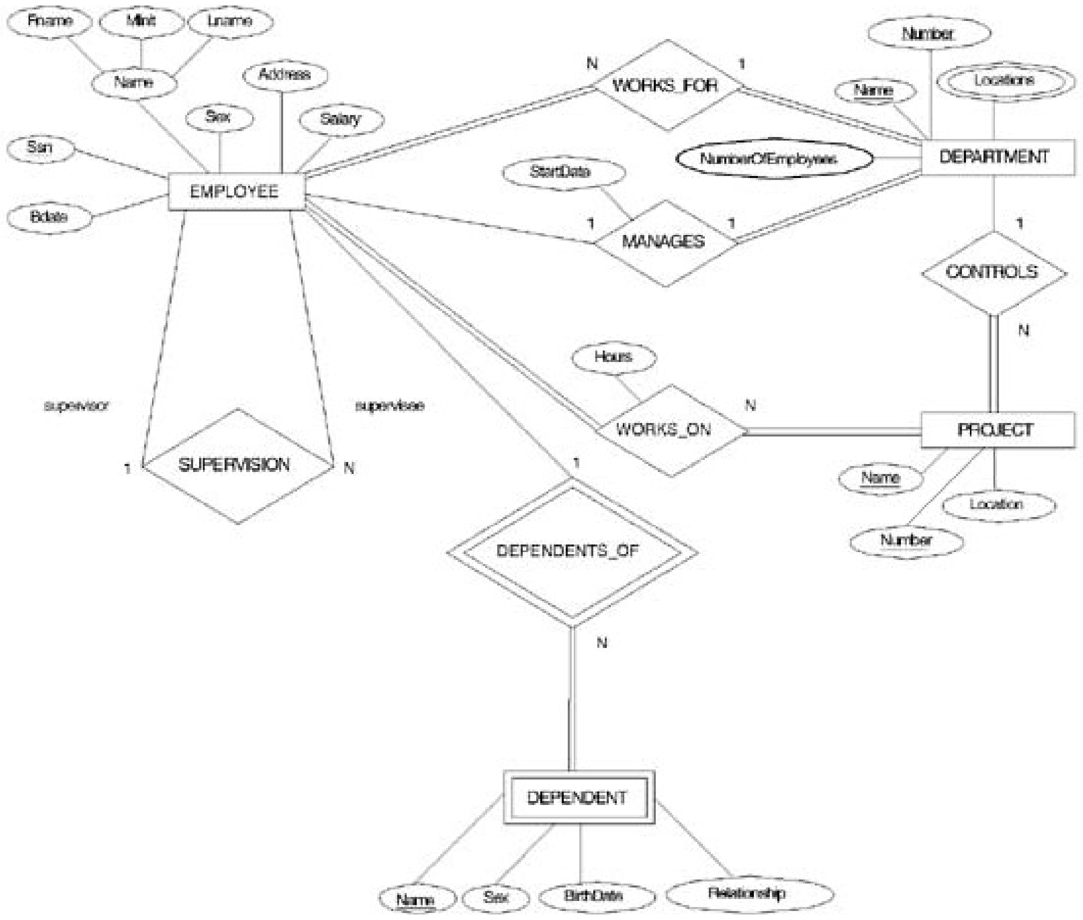
Example relationship instances of the WORKS_FOR relationship between EMPLOYEE and DEPARTMENT



Example relationship instances of the WORKS_ON relationship between EMPLOYEE and PROJECT



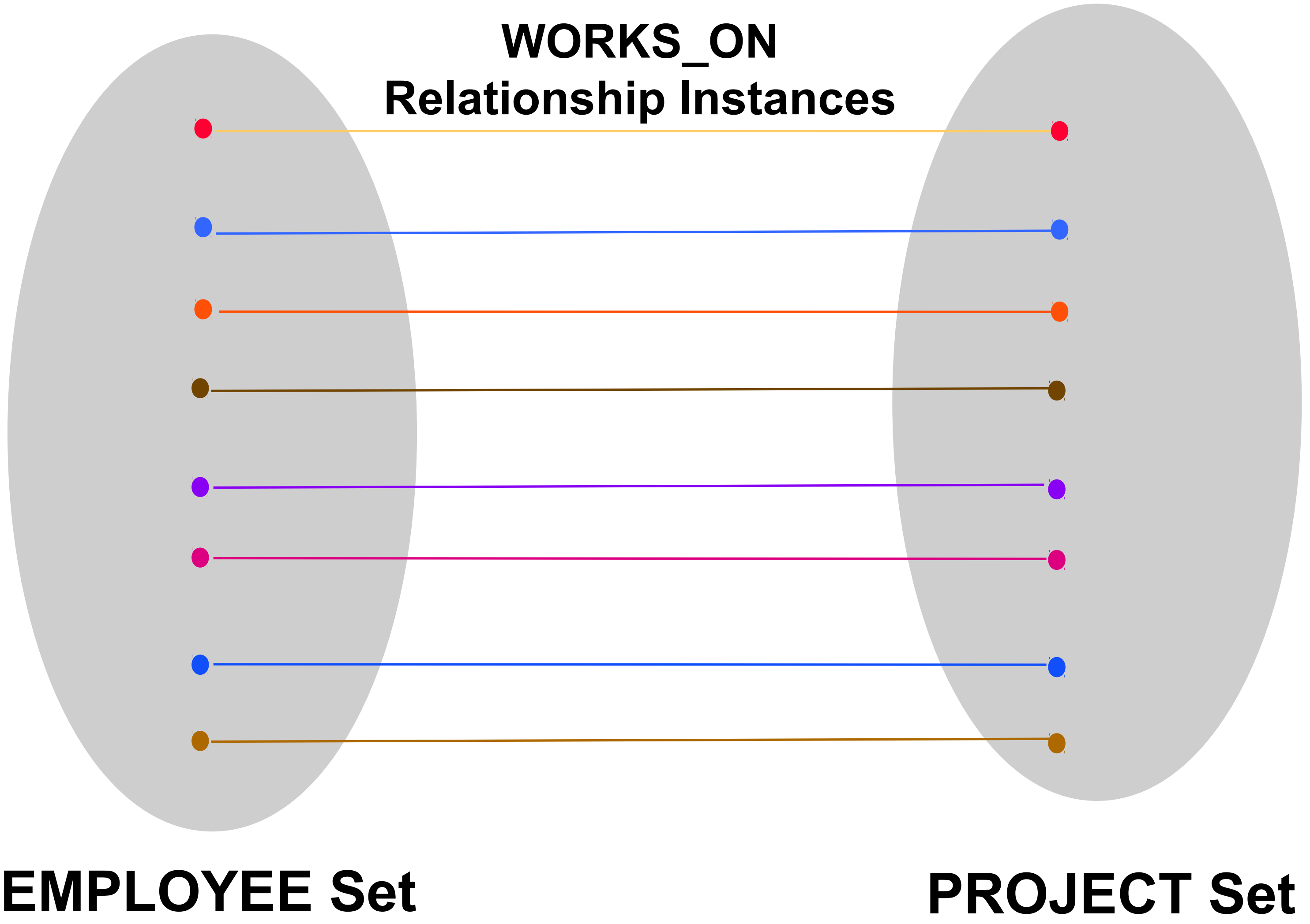
ER DIAGRAM – Relationship Types are: WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



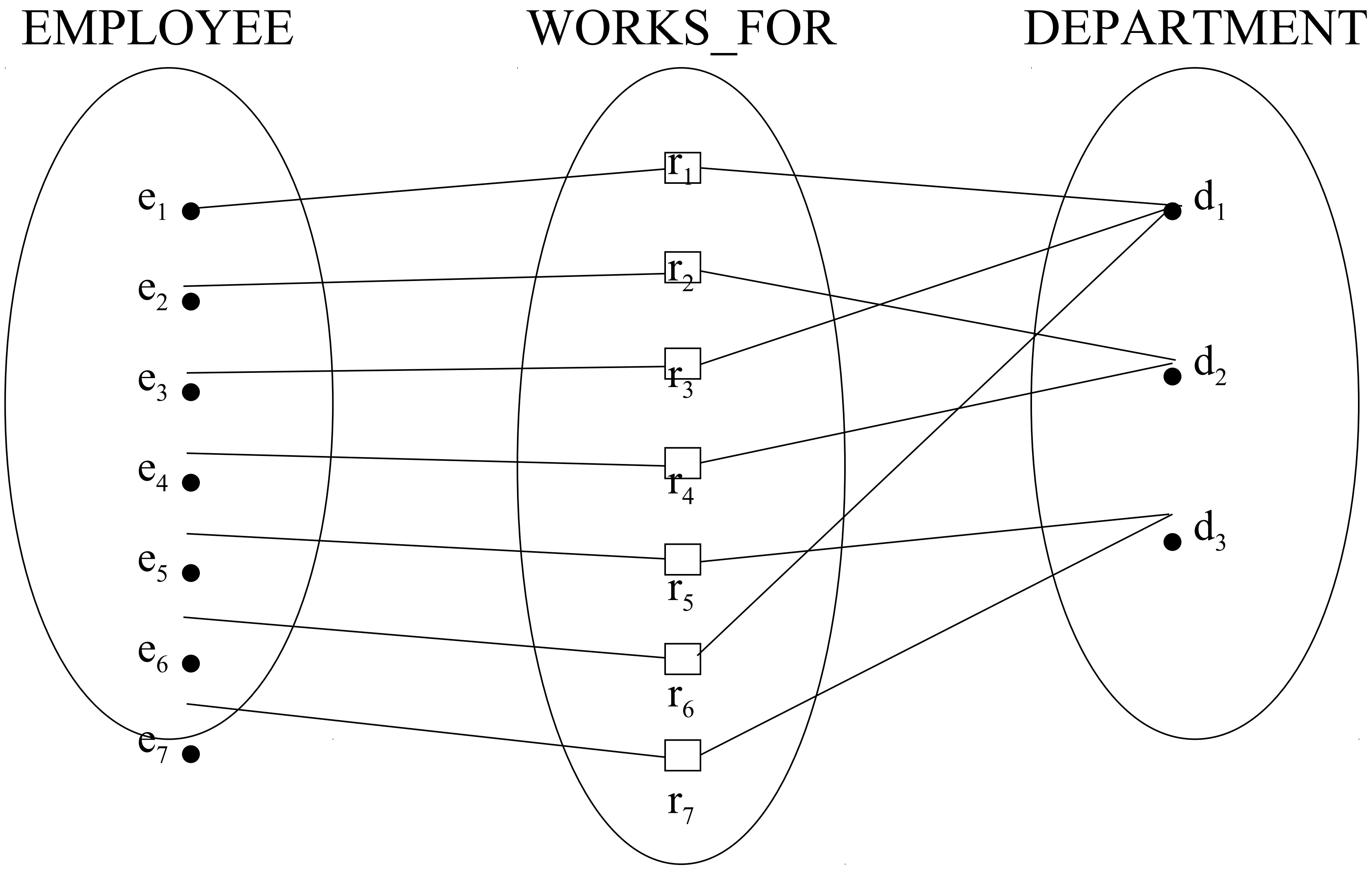
Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Maximum Cardinality
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many
  - Minimum Cardinality (also called participation constraint or existence dependency constraints)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory, existence-dependent)

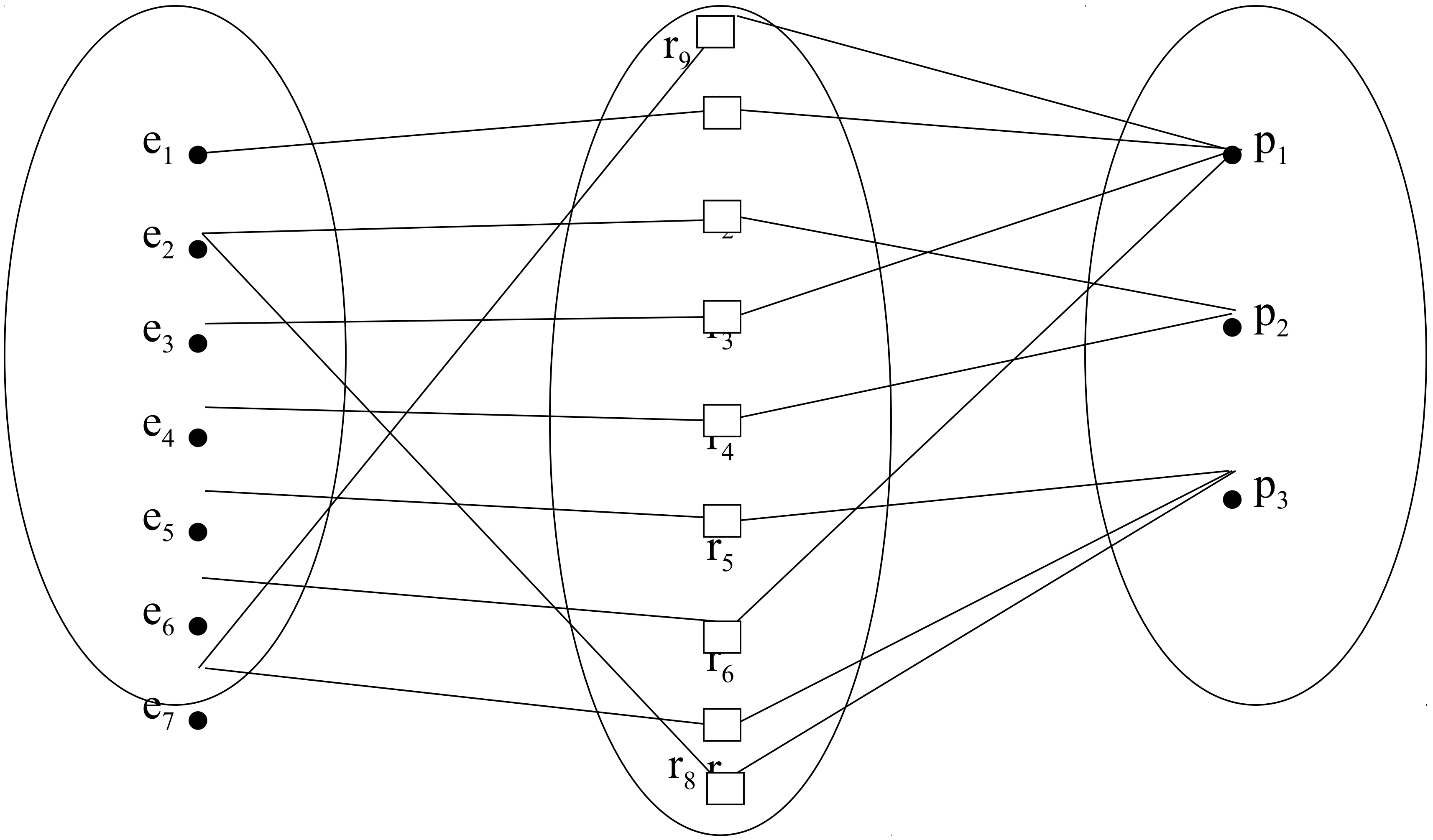
One-to-One WORKS_ON Relationship



Many-to-one (N:1) RELATIONSHIP



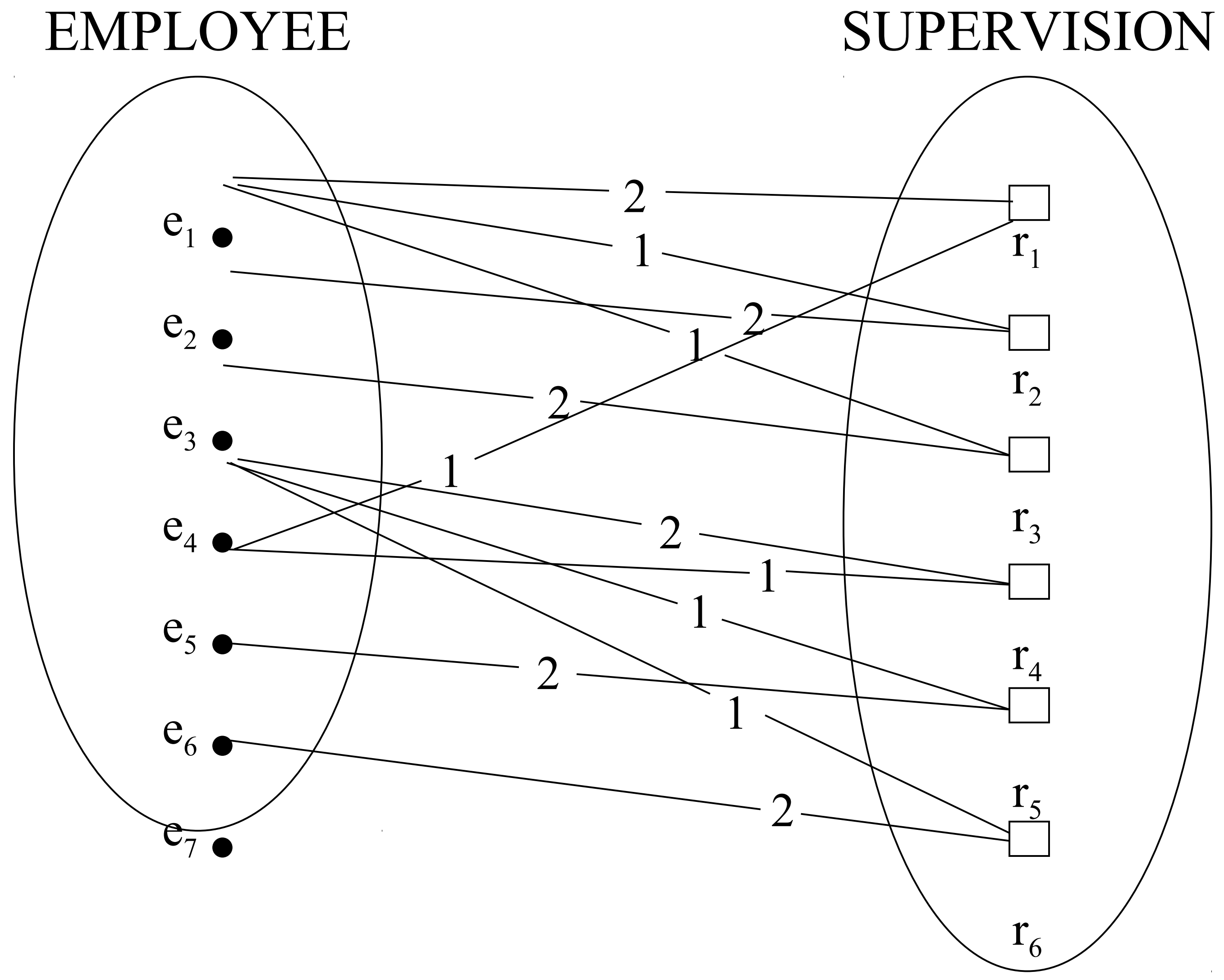
Many-to-many (M:N) RELATIONSHIP



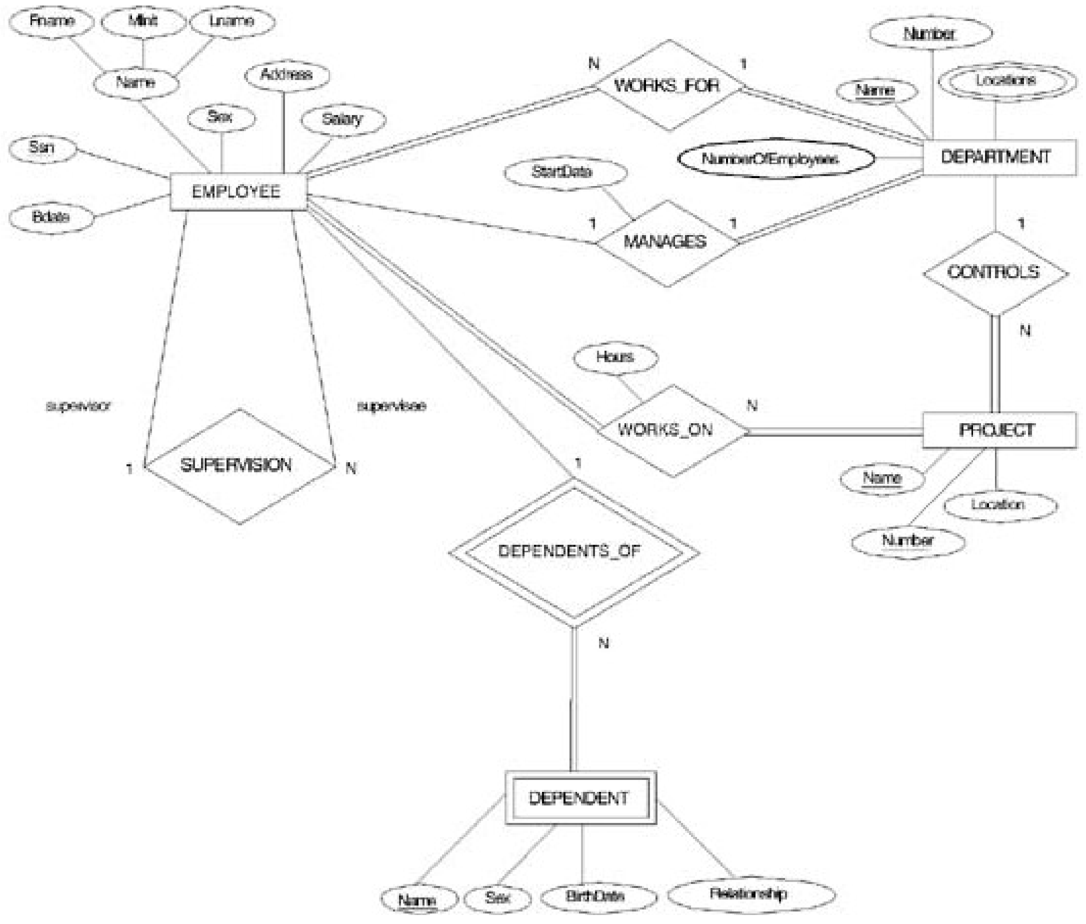
Recursive Relationship

- We can also have a **recursive** relationship type.
- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, need to display role names to distinguish participations.

A RECURSIVE RELATIONSHIP SUPERVISION



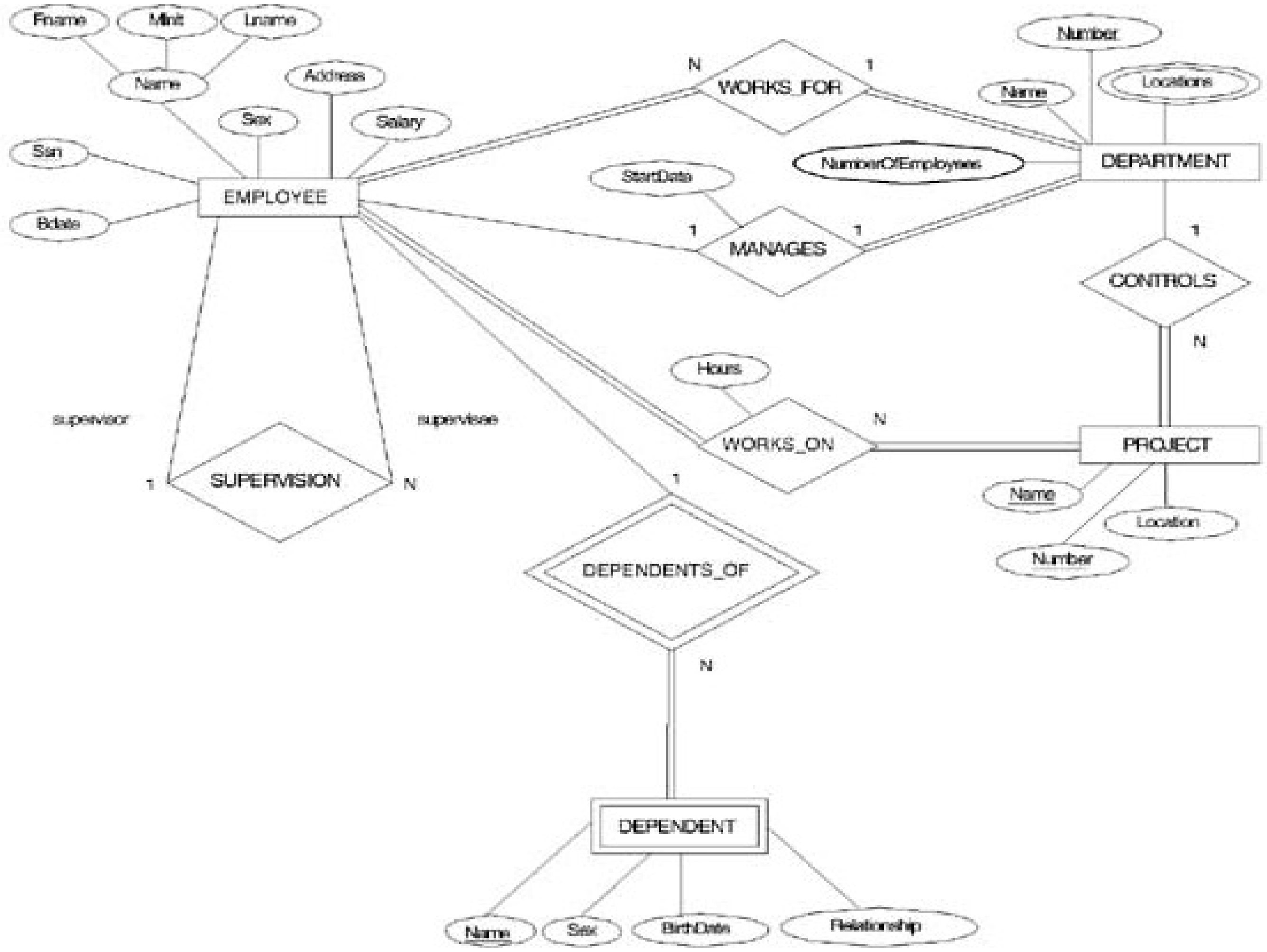
Recursive Relationship Type is: SUPERVISION (participation role names are shown)



Attributes of Relationship types

- A relationship type can have attributes;
- For example, HoursPerWeek of WORKS_ON; its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

Attribute of a Relationship Type is: Hours of WORKS_ON



Specifying Relationship

- *Examine every combination of two entities and see whether there is a possible relationship between them*
  - *This is often documented using a matrix that lists the entity names on both axes.*
  - *A symbol is entered at the intersection of each row and column to indicate the existence of a possible relationship.*
  - *This technique becomes unusable if the model is large.*
- *Look at the requirements documents to find relationships indicated by the documents.*
- *The two techniques can be used together*

Structural Constraints – one way to express semantics of relationships

Structural constraints on relationships:

- **Cardinality ratio** (of a binary relationship): 1:1, 1:N, N:1, or M:N

  SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK.

- **Participation constraint** (on each participating entity type): total (called *existence dependency*) or partial.

  SHOWN BY DOUBLE LINING THE LINK

NOTE: These are easy to specify for Binary Relationship Types.

Alternative (min, max) notation for relationship structural constraints:

- 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 the knowledge of mini-world constraints

Examples:

- A department has *exactly one* manager and an employee can manage *at most one* department.

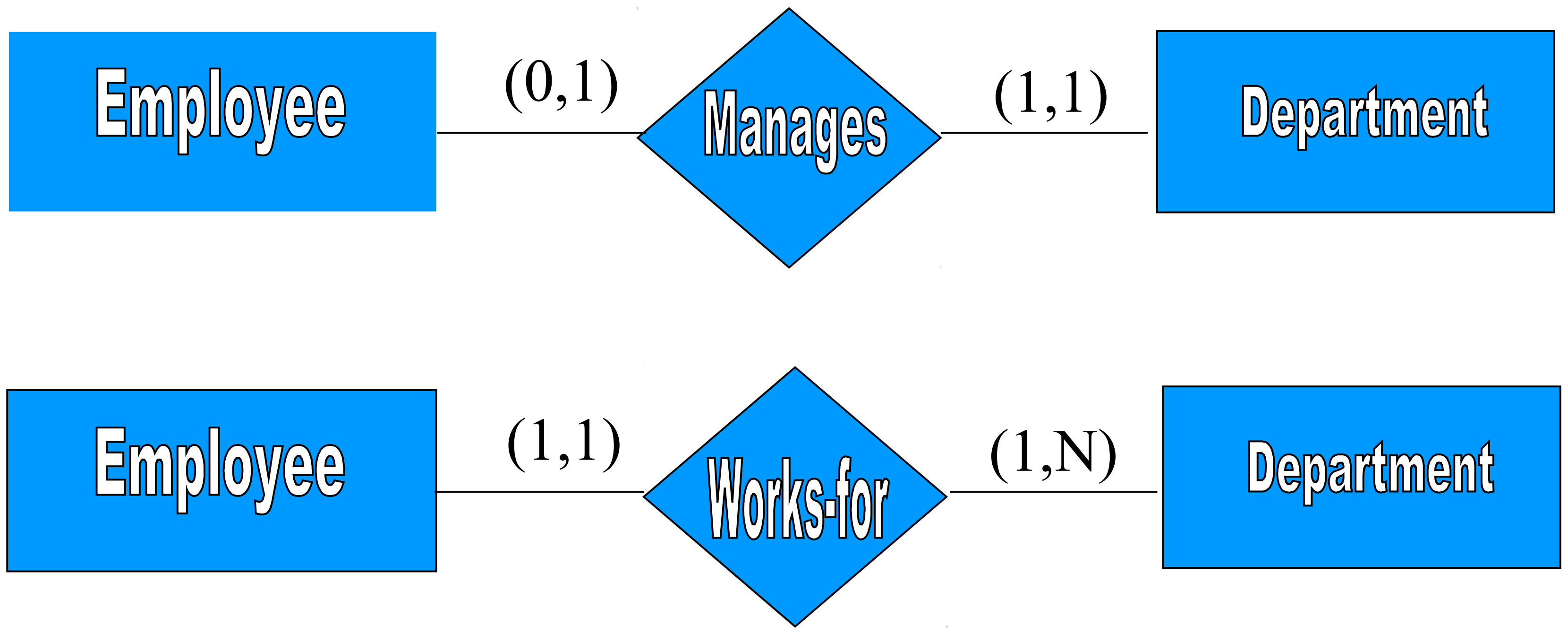
  Specify (0,1) for participation of EMPLOYEE in MANAGES

  Specify (1,1) for participation of DEPARTMENT in MANAGES
- An employee can work for *exactly one* department but a department can have *any number of employees*.

  Specify (1,1) for participation of EMPLOYEE in WORKS_FOR

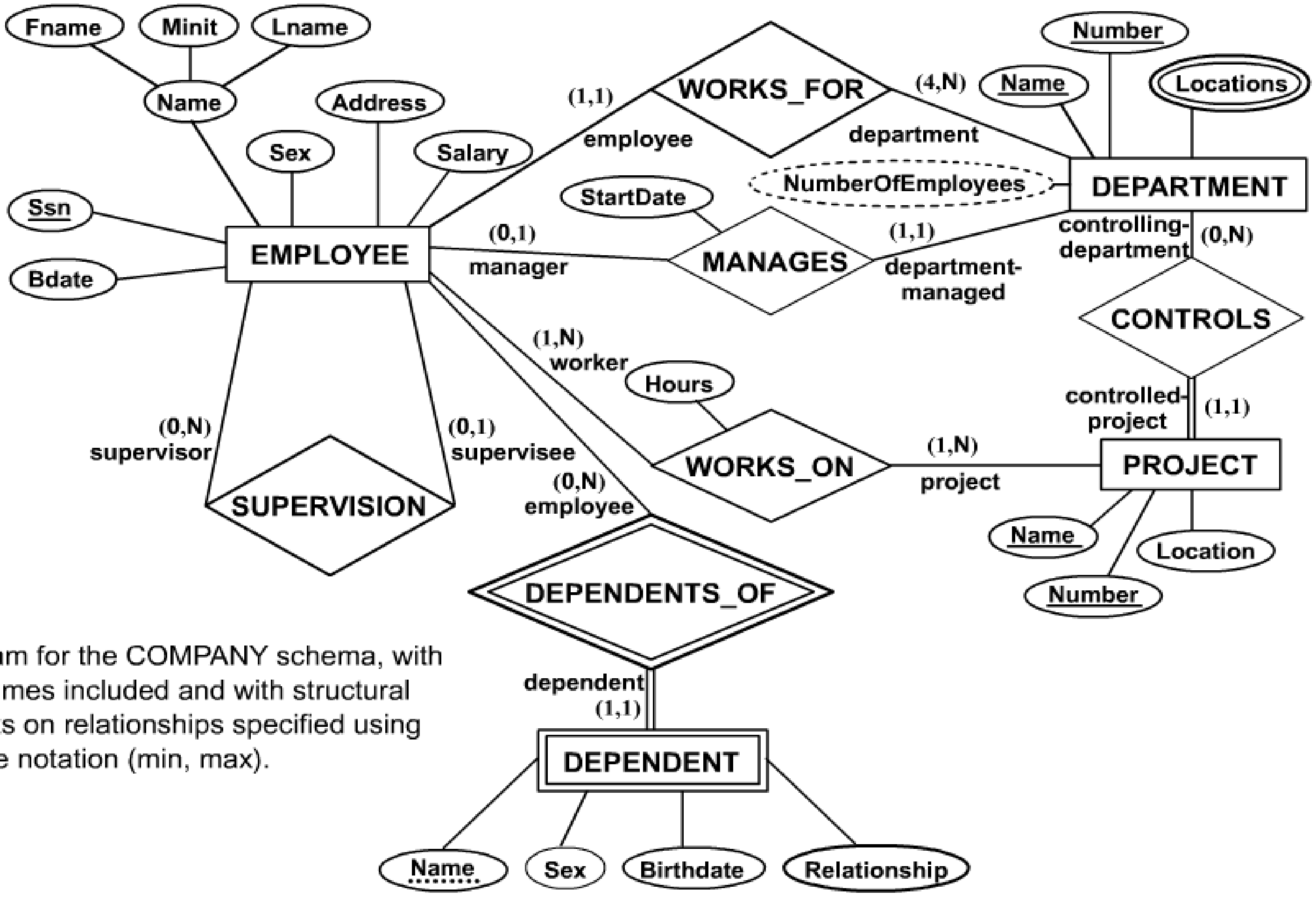
  Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation relationship constraints



COMPANY ER Schema Diagram using (min, max) notation

Alternative ER Notations



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

Naming Convention

- Choose names (for entity types, attributes, relationship types) that convey the meanings attached to the different constructs in the schema
- Use singular names for entity types
- Use uppercase for entity type and relation type names
- Use capitalized name for attributes
- Use lowercase for role names
- Nouns tend to give rise to entity type names
- Verbs tend to indicate names of relationship types

Design Choices for ER

- Design process: iterative process with some common refinements:
  - A concept may be first modeled as an attribute and then refined into relationship because it is determined that the attribute is a reference to another entity types
  - An attribute that exist in several entity types may be promoted to an independent entity types
    - In UNIVERSITY db, STUDENT, INSTRUCTOR & COURSE, each has an attribute Department in the initial design, then create DEPARTMENT entity with DeptName as attribute and relate it to the three entities. Other attribute may be discovered later.
  - An inverse refinement to the previous case may be applied
    - If DEPARTMENT exist in the initial design with single attribute and relate only to one other entity, STUDENT. Then reduced it into attribute of STUDENT.
  - Refinement on specialization, generalization & higher degree relationship

Problem ER Notation

- THE ENTITY RELATIONSHIP MODEL IN ITS ORIGINAL FORM DID NOT SUPPORT THE SPECIALIZATION/ GENERALIZATION ABSTRACTIONS

Data Modeling Tools

A number of popular tools that cover conceptual modeling and mapping into relational schema design. Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, etc.

POSITIVES: serves as documentation of application requirements, easy user interface - mostly graphics editor support

Problems with Current Modeling Tools

- DIAGRAMMING
  - Poor conceptual meaningful notation.
  - To avoid the problem of layout algorithms and aesthetics of diagrams, they prefer boxes and lines and do nothing more than represent (primary-foreign key) relationships among resulting tables.(a few exceptions)
- METHODOLGY
  - lack of built-in methodology support.
  - poor tradeoff analysis or user-driven design preferences.
  - poor design verification and suggestions for improvement.

Some of the Currently Available Automated Database Design Tools

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration and space and security management
Oracle	Developer 2000 and Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum Technology	Platinum Enterprice Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational	Rational Rose	Modeling in UML and application generation in C++ and JAVA
Rogue Ware	RW Metro	Mapping from O-O to relational model
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design and reengineering Visual Basic and Visual C++