

Entity-Relationship (ER) Model

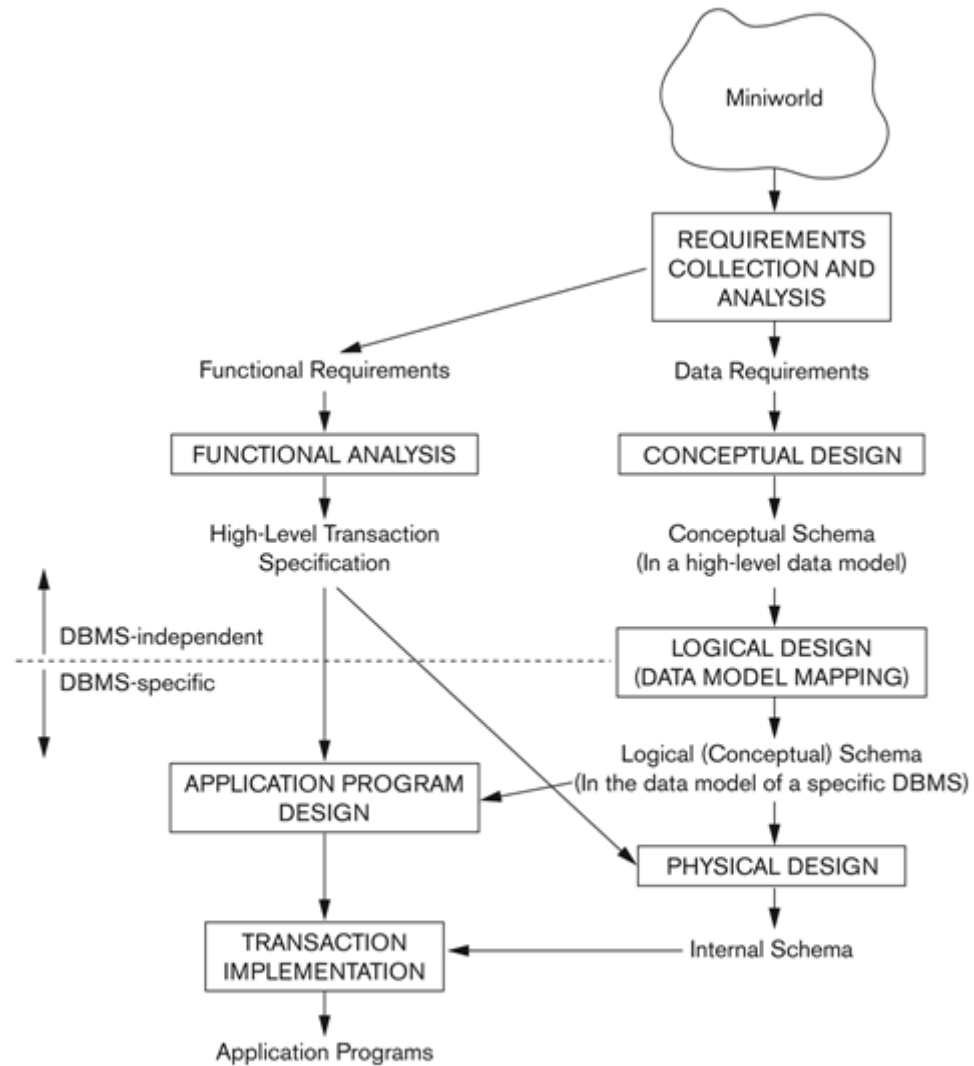
- Database Design Process
- Example Database Application (COMPANY)
- ER Model Concepts
 - Entities and Attributes
 - Entity Types, Value Sets, and Key Attributes
 - Relationships and Relationship Types
 - Weak Entity Types
 - Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram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

Source: Fundamentals of Database Systems by Ramez Elmasri and Shamkant B. Navathe

Database Design Process

- Two main activities:
 - Database design
 - Applications design
- Database design
 - To design the conceptual schema for a database application
- Applications design focuses on the programs and interfaces that access the database
 - Generally considered part of software engineering

Database Design Process



Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of 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 of the department manager. A department may have several locations.
 - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database (Contd.)

- 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 the employee.

ER Model Concepts

- Usually a noun in singular
- Represented by a rectangle with a label
- First letter in capitals

- 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

Name

- Attributes are properties used to describe an entity.
 - For example an EMPLOYEE entity may have the attributes 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, ...

Employee

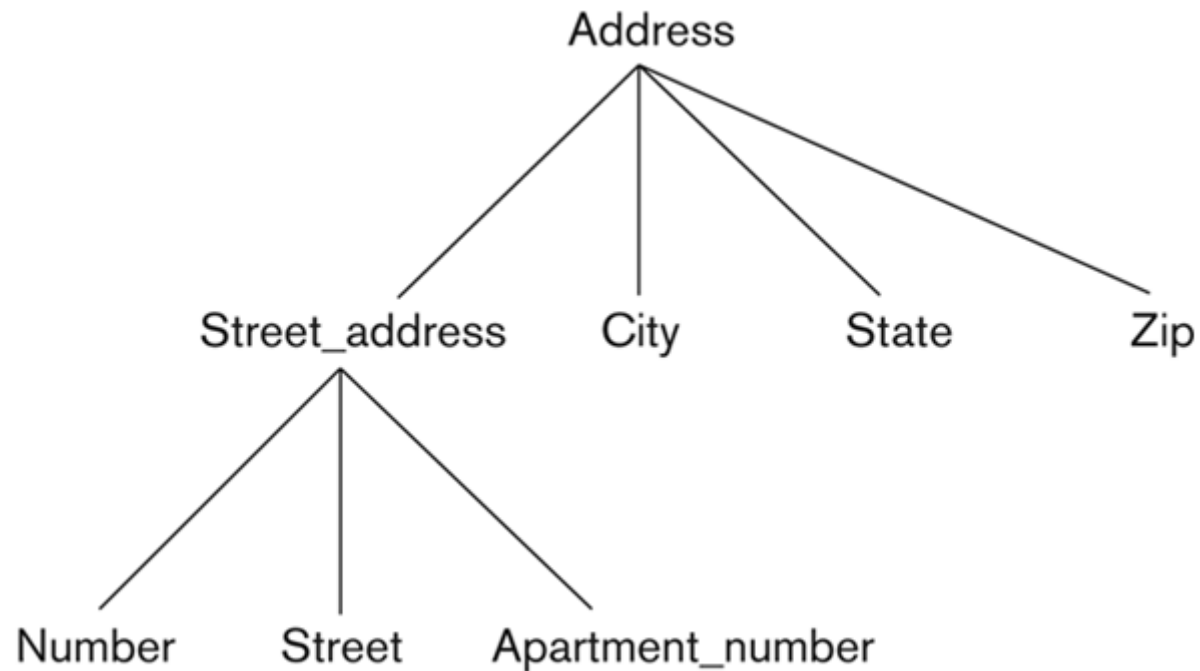
Types of Attributes (1)

- 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)}
 - Multiple PreviousDegrees values can exist
 - Each has four subcomponent attributes:
 - College, Year, Degree, Field

Example of a composite attribute



Entity Types and Key Attributes (1)

- Entities with the same basic attributes are grouped or typed into an entity type.
 - For example, the entity type EMPLOYEE and PROJECT.
- 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, SSN of EMPLOYEE.

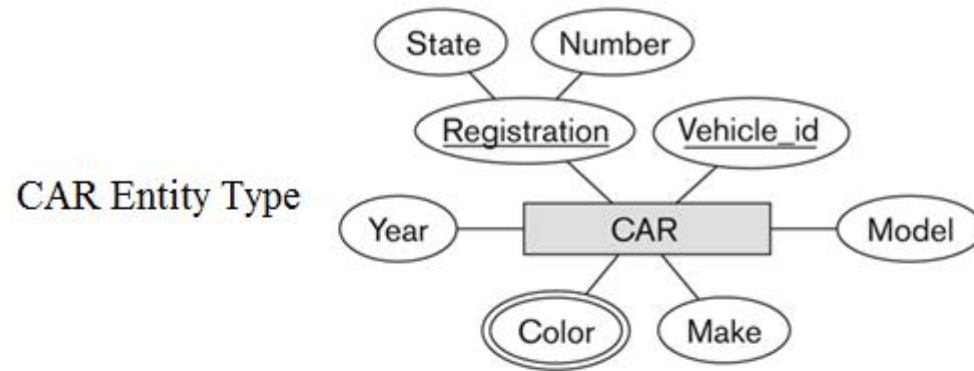
Entity Types and Key Attributes (2)

- A key attribute may be composite.
 - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
 - The CAR entity type may have two keys:
 - VehicleIdentificationNumber (popularly called VIN)
 - VehicleTagNumber (Number, State), aka license plate number.
- Each key is underlined

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
 - Each attribute is connected to its entity type
 - Components of a composite attribute are connected to the oval representing the composite attribute
 - Each key attribute is underlined
 - Multivalued attributes displayed in double ovals
- See CAR example on next slide

Entity Type CAR with two keys and a corresponding Entity Set



CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

Entity set
with three
entities

CAR₁
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR₃
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

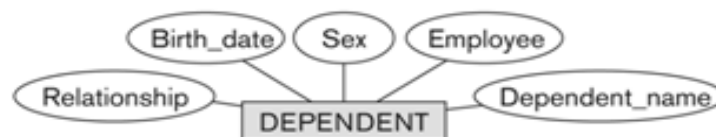
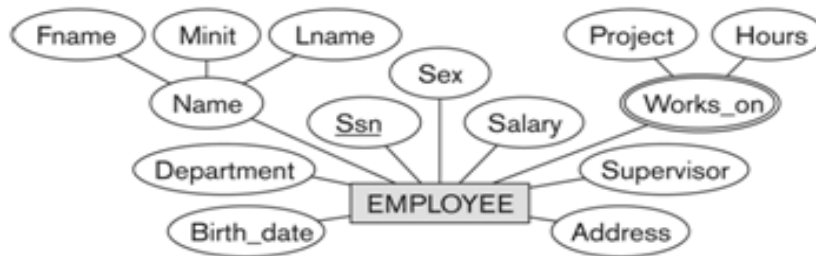
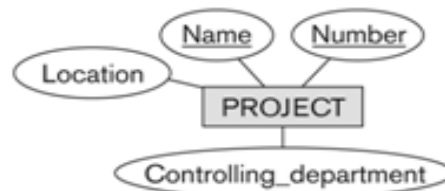
Entity Set

- Each entity type will have a collection of entities stored in the database
 - Called the **entity set**
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- Entity set is the current *state* of the entities of that type that are stored in the database

Initial Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
 - DEPARTMENT
 - PROJECT
 - EMPLOYEE
 - DEPENDENT
- Their initial design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT



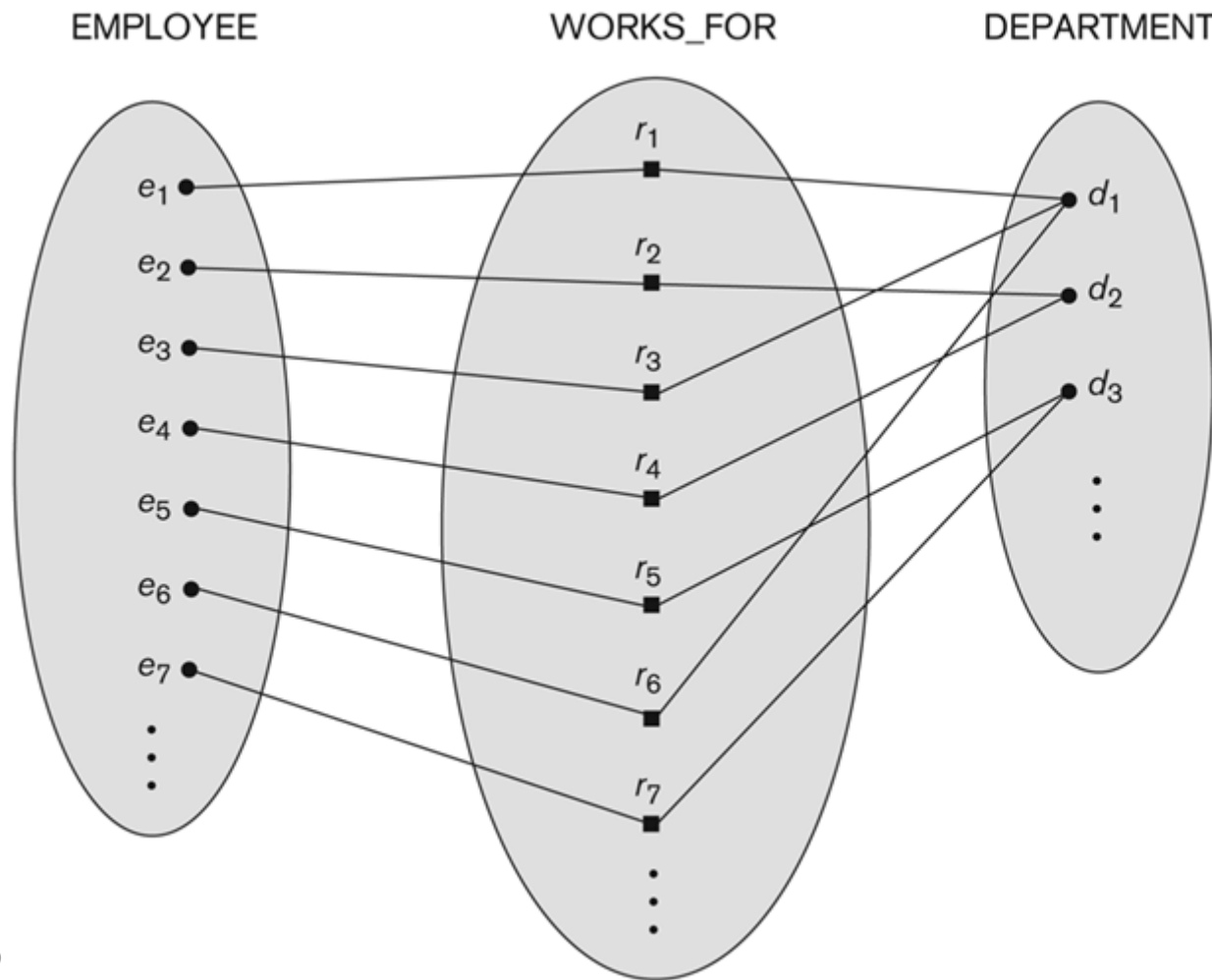
Refining the initial design by introducing **relationships**

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
 - Entities (and their entity types and entity sets)
 - Attributes (simple, composite, multivalued)
 - Relationships (and their relationship types and relationship sets)

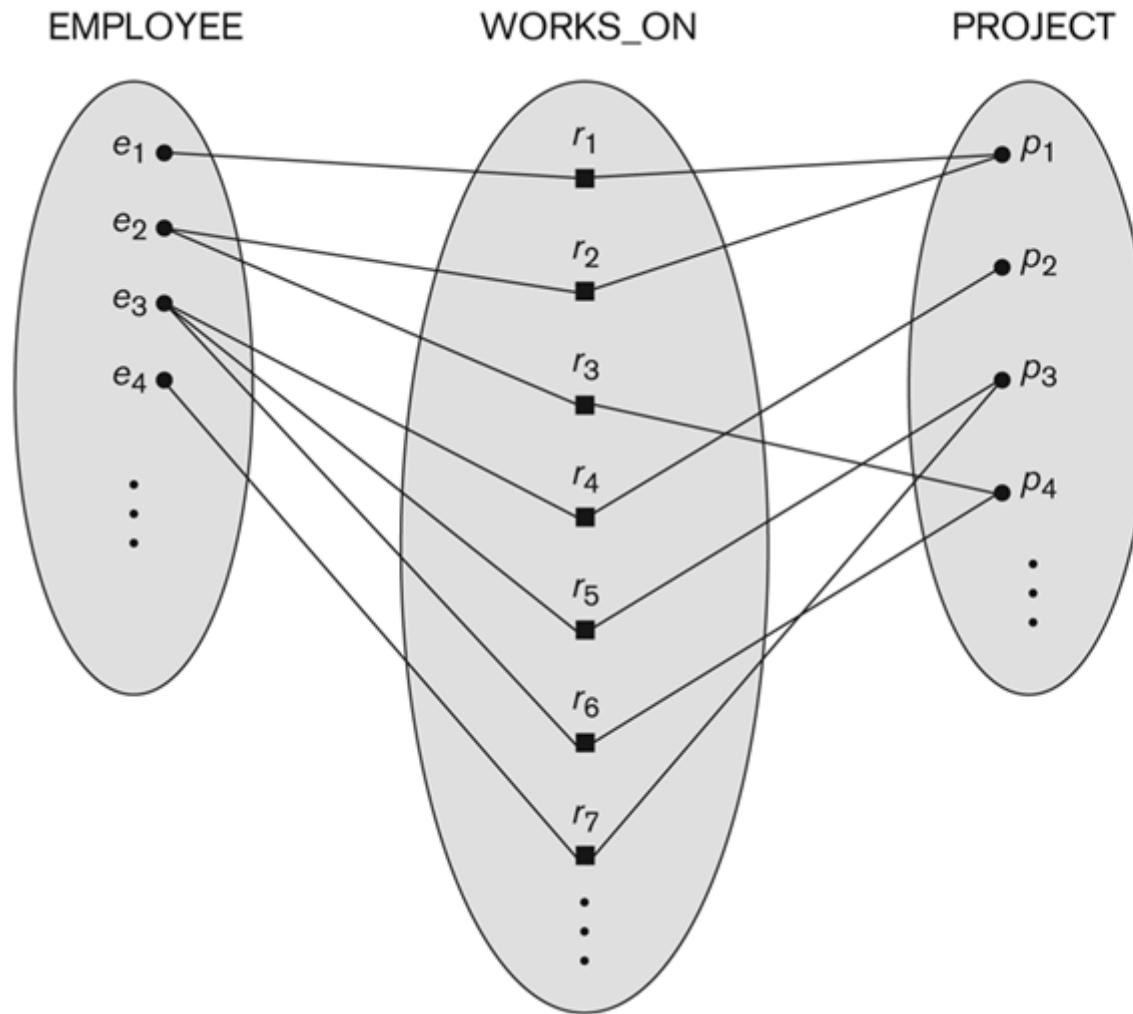
Relationships and Relationship Types (1)

- 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.

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT



Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT



Relationship type vs. relationship set (1)

- Relationship Type:
 - Is the schema description of a relationship
 - Identifies the relationship name and the participating entity types
 - Also identifies certain relationship constraints
- Relationship Set:
 - The current set of relationship instances represented in the database
 - The current *state* of a relationship type

Relationship type vs. relationship set (2)

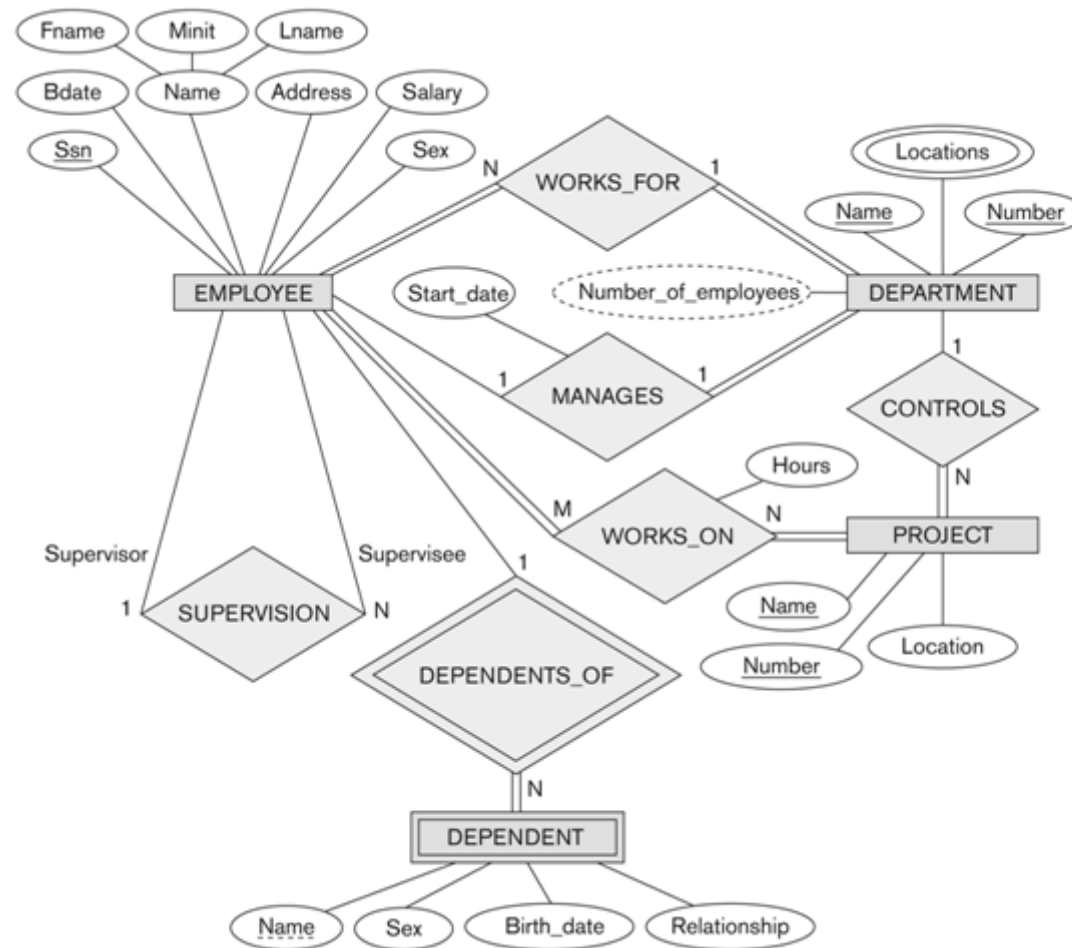
- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
 - Diamond-shaped box is used to display a relationship type
 - Connected to the participating entity types via straight lines

Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships(degree 2)
- Listed below with their participating entity types:
 - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
 - MANAGES (also between EMPLOYEE, DEPARTMENT)
 - CONTROLS (between DEPARTMENT, PROJECT)
 - WORKS_ON (between EMPLOYEE, PROJECT)
 - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
 - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



Discussion on Relationship Types

- In the refined design, some attributes from the initial entity types are refined into relationships:
 - Manager of DEPARTMENT -> MANAGES
 - Works_on of EMPLOYEE -> WORKS_ON
 - Department of EMPLOYEE -> WORKS_FOR
 - etc
- In general, more than one relationship type can exist between the same participating entity types
 - MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
 - Different meanings and different relationship instances.

Recursive Relationship Type

- An relationship type whose with the same participating entity type in **distinct roles**
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
 - supervisor (or boss) role
 - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
 - One employee in *supervisor* role
 - One employee in *supervisee* role

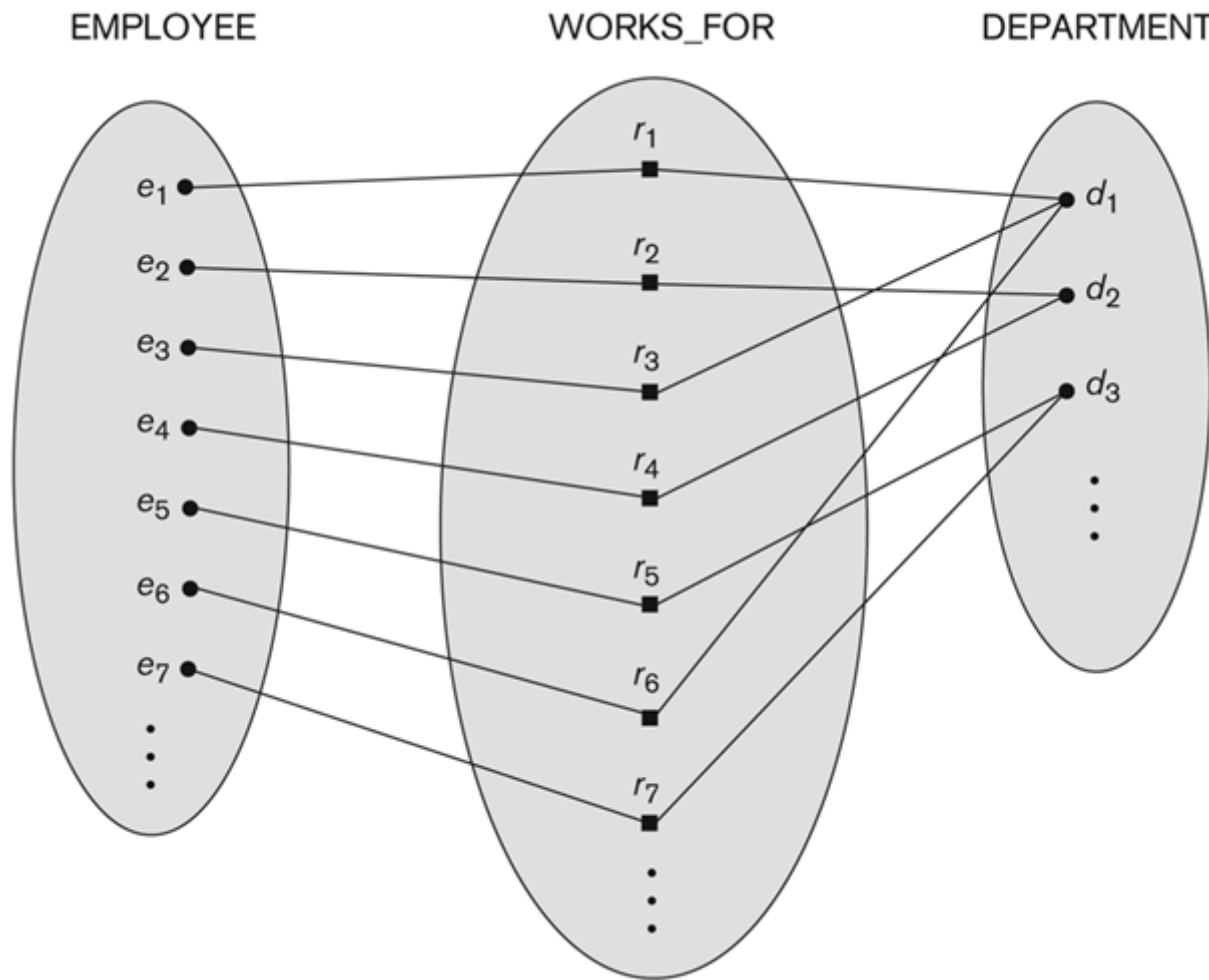
Weak Entity Types

- 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:**
 - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
 - Name of DEPENDENT is the *partial key*
 - DEPENDENT is a *weak entity type*
 - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

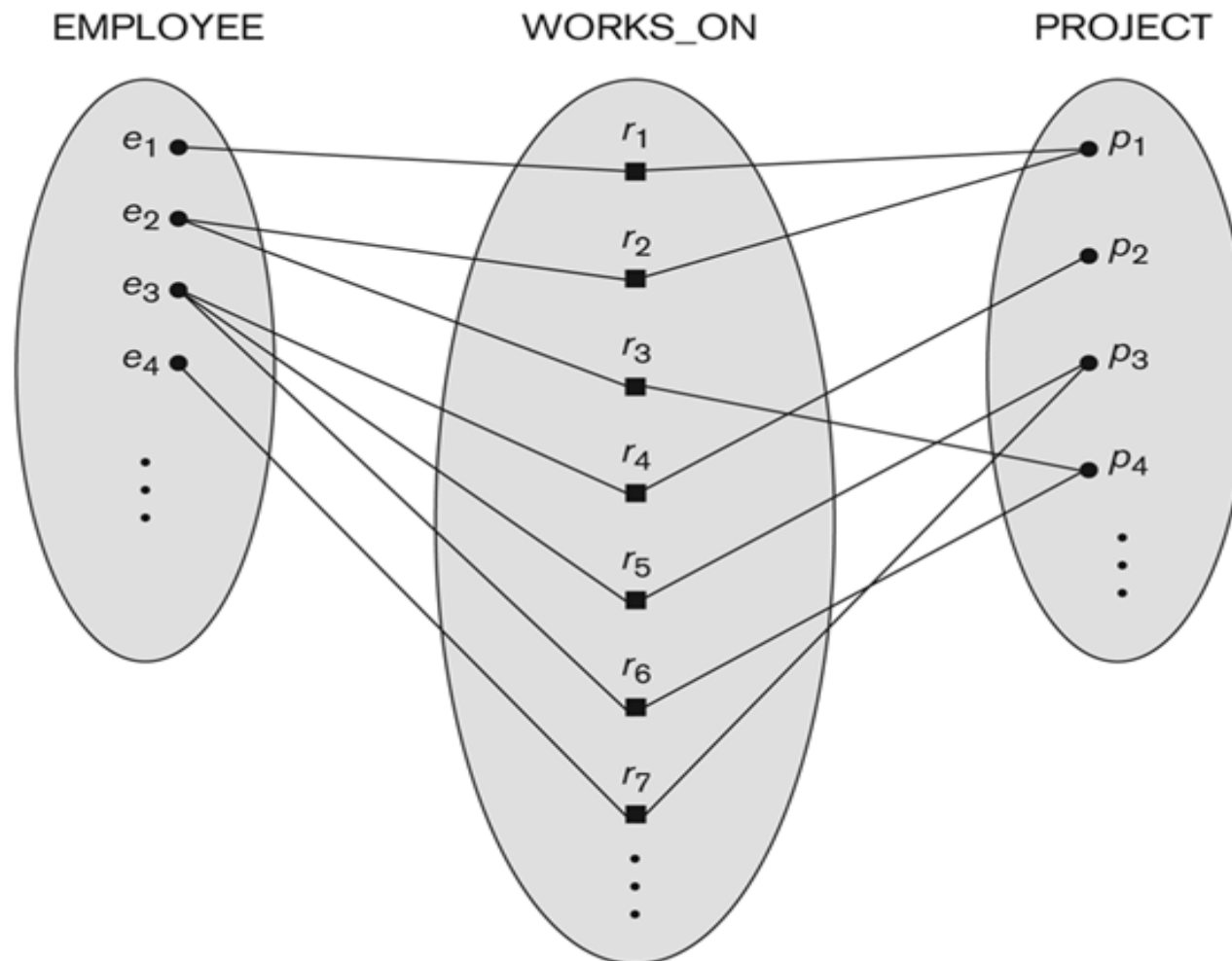
Constraints on Relationships

- Constraints on Relationship Types
 - (Also known as ratio constraints)
 - Cardinality Ratio (specifies *maximum* participation)
 - One-to-one (1:1)
 - One-to-many (1:N) or Many-to-one (N:1)
 - Many-to-many (M:N)
 - Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
 - zero (optional participation, not existence-dependent)
 - one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship



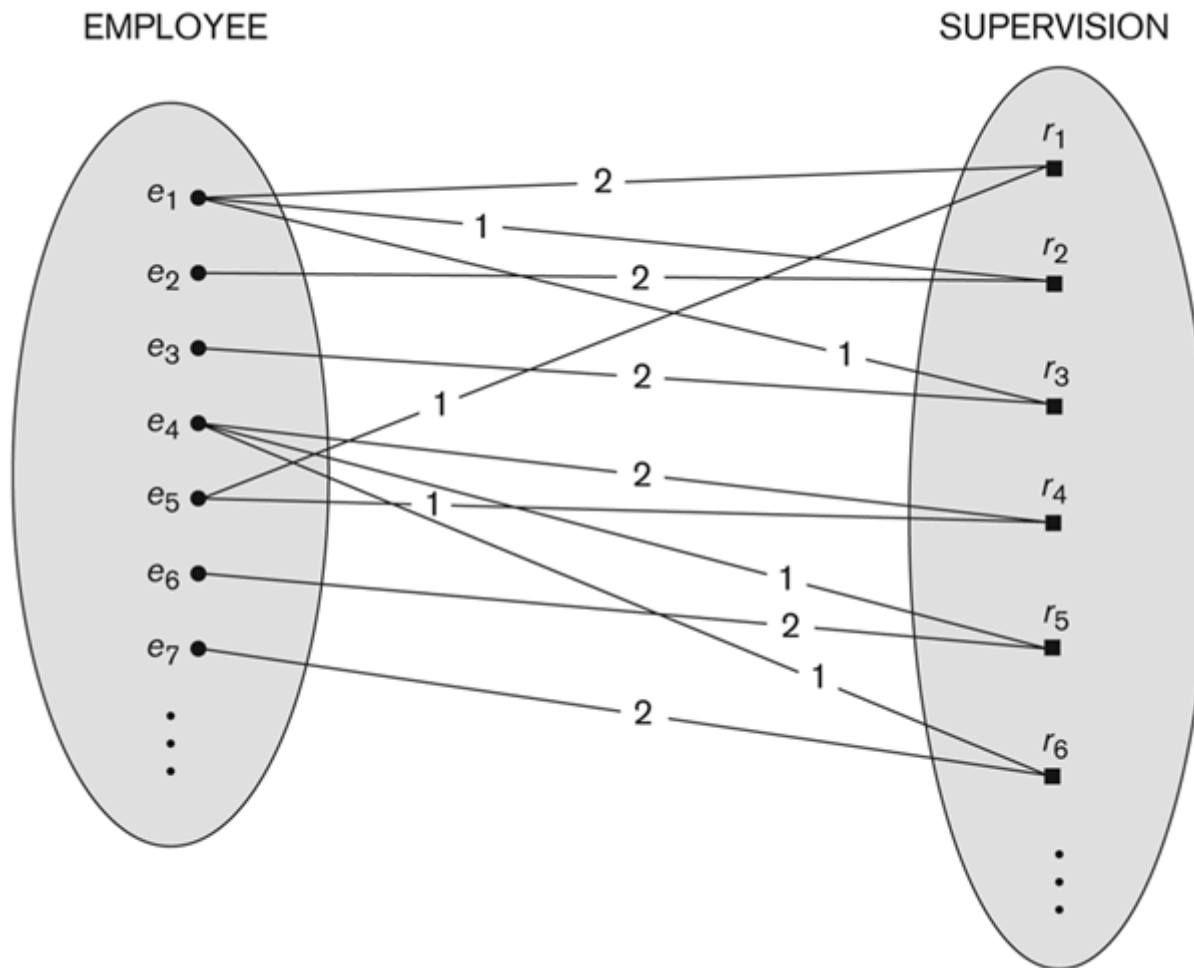
Many-to-many (M:N) Relationship



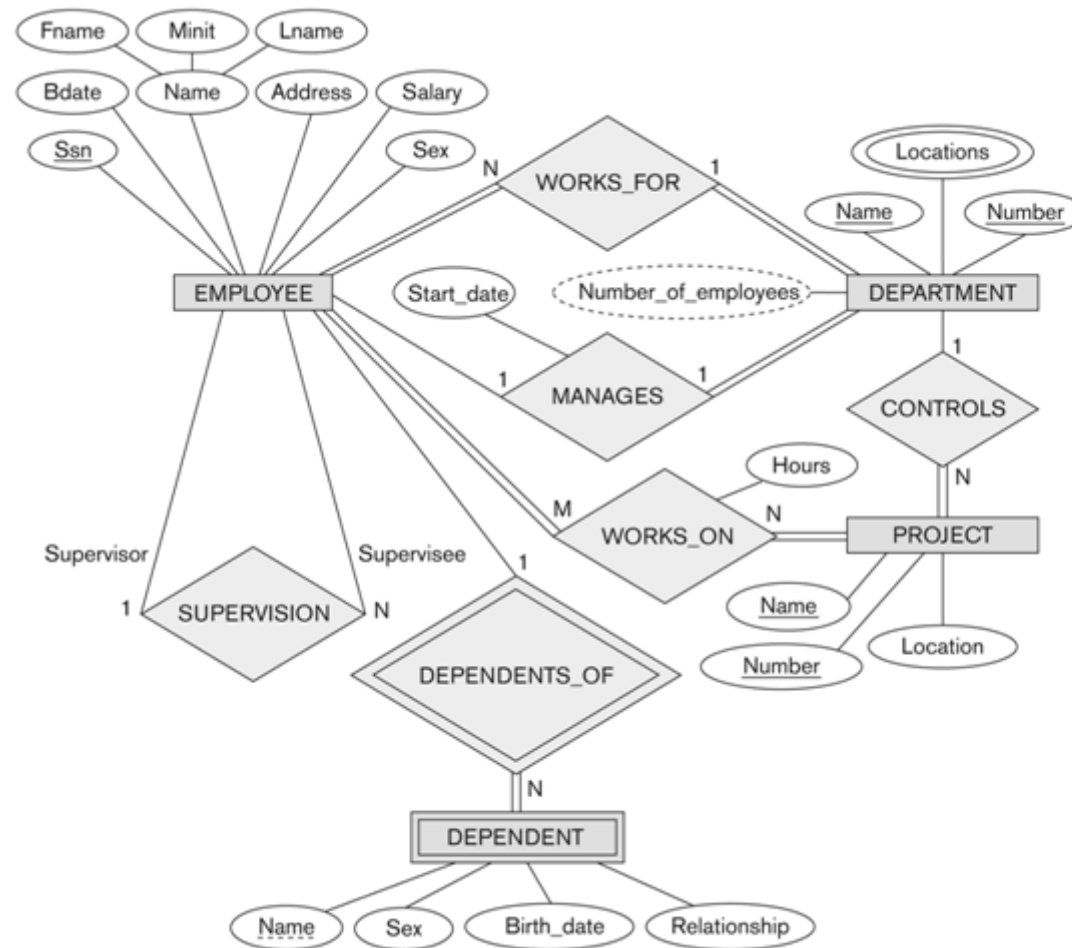
Displaying a recursive relationship

- In 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`



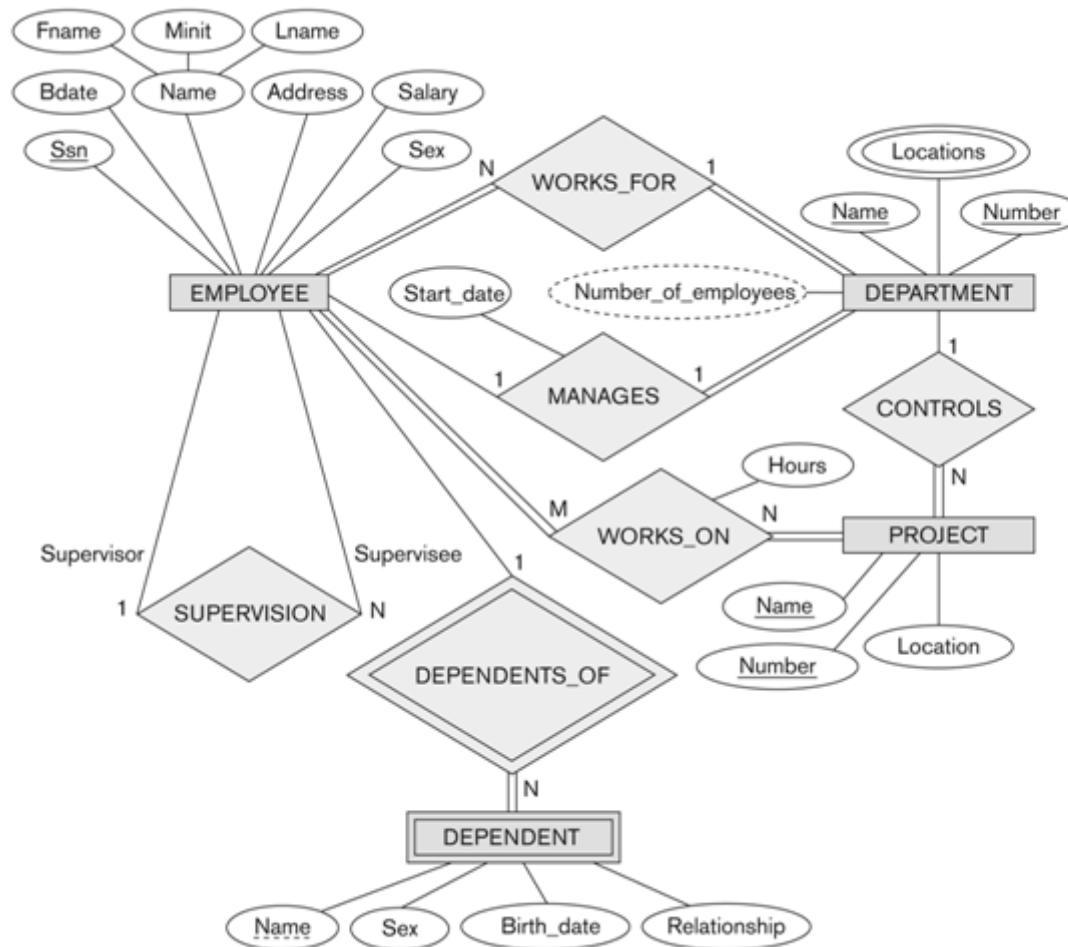
Recursive Relationship Type is: SUPERVISION (participation role names are shown)



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.
 - A value of HoursPerWeek depends on a particular (employee, project) combination
 - Most relationship attributes are used with M:N relationships
 - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

Example Attribute of a Relationship Type: Hours of WORKS_ON



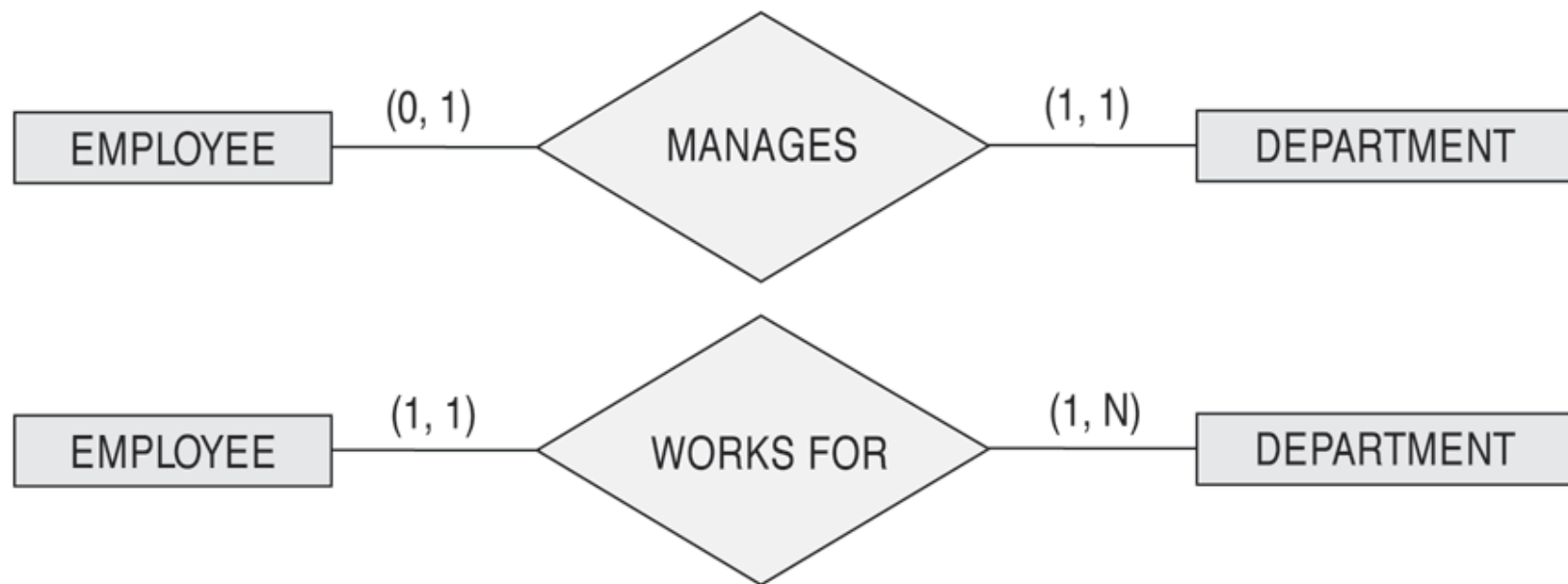
Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
 - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
 - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

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

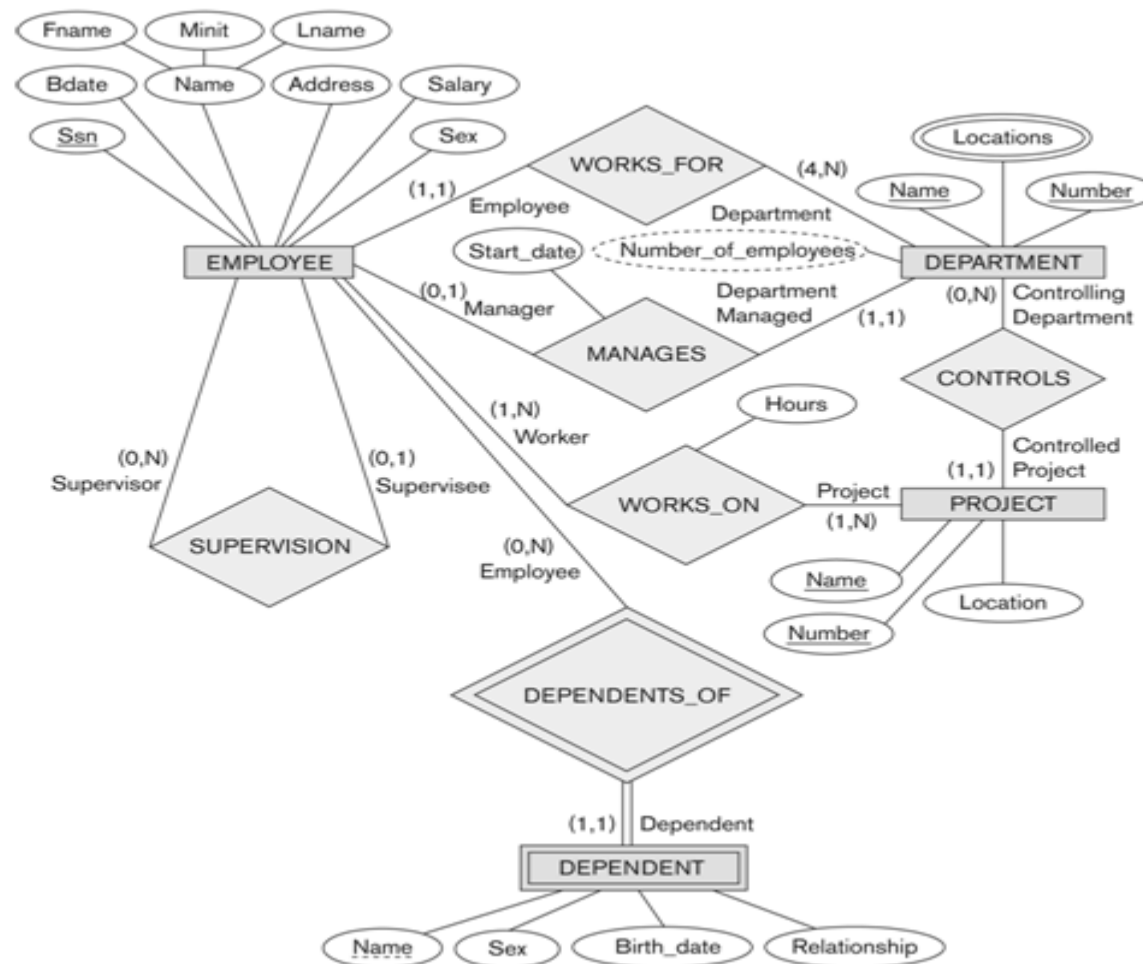
- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have $\text{min} \leq \text{max}$, $\text{min} \geq 0$, $\text{max} \geq 1$
- 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 for relationship constraints



Read the min,max numbers next to the entity type and looking **away from** the entity type

COMPANY ER Schema Diagram using (min, max) notation








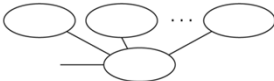

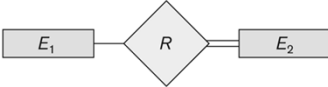

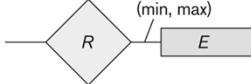


Alternative diagrammatic notation

- ER diagrams is one popular example for displaying database schemas
- Many other notations exist in the literature and in various database design and modeling tools
- Appendix A illustrates some of the alternative notations that have been used
- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

Summary of notation for ER diagrams

Figure 3.14
Summary of the
notation for ER
diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	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

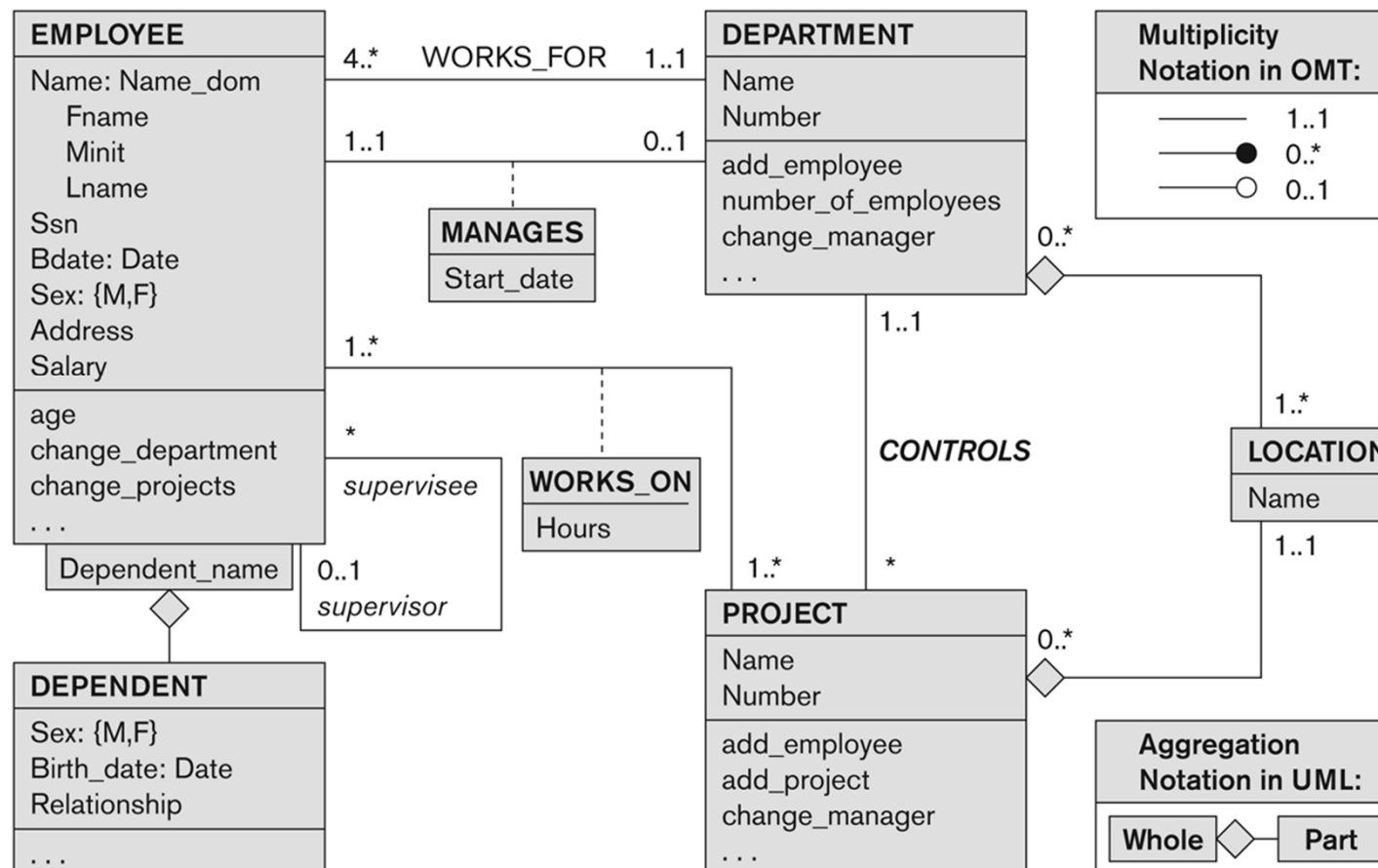
UML class diagrams

- Represent classes (similar to entity types) as large rounded boxes with three sections:
 - Top section includes entity type (class) name
 - Second section includes attributes
 - Third section includes class operations (operations are not in basic ER model)
- Relationships (called associations) represented as lines connecting the classes
 - Other UML terminology also differs from ER terminology
- Used in database design and object-oriented software design
- UML has many other types of diagrams for software design (see Chapter 12)

UML class diagram for COMPANY database schema

Figure 3.16

The COMPANY conceptual schema in UML class diagram notation.



Other alternative diagrammatic notations

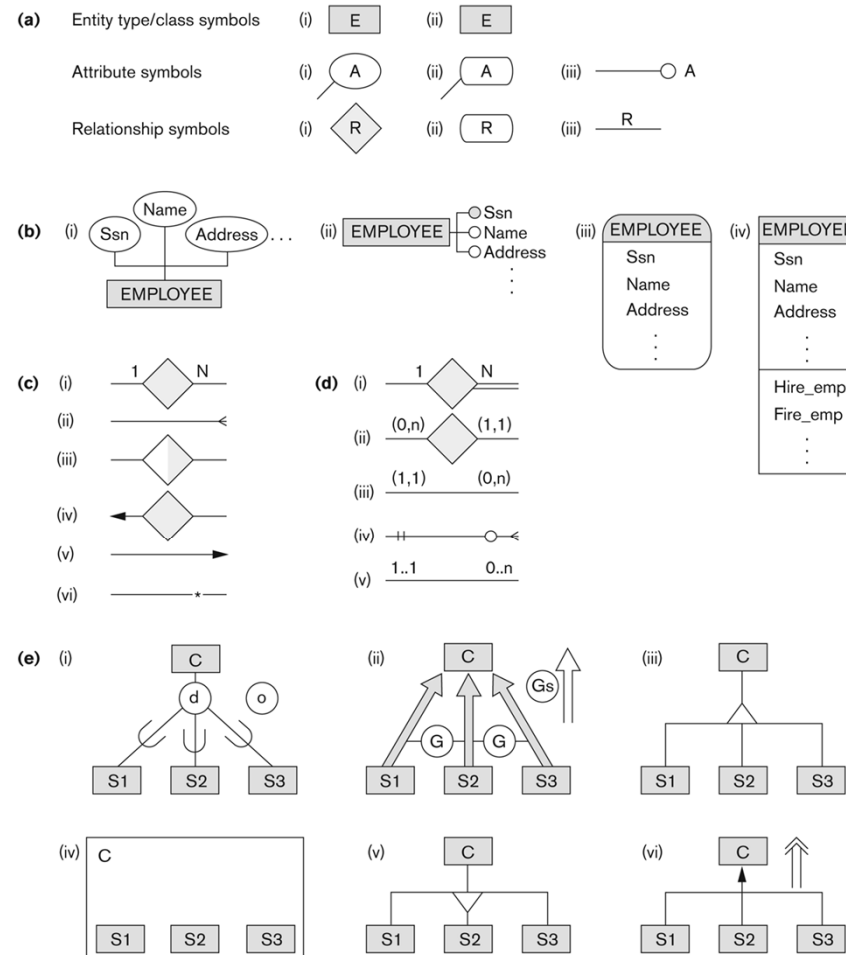


Figure A.1

Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

Relationships of Higher Degree

- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary and of degree n are called n -ary
- In general, an n -ary relationship is not equivalent to n binary relationships
- Constraints are harder to specify for higher-degree relationships ($n > 2$) than for binary relationships

Discussion of n-ary relationships ($n > 2$)

- In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on next slide)
- If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)
- In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Figure 3.17c)

Example of a ternary relationship

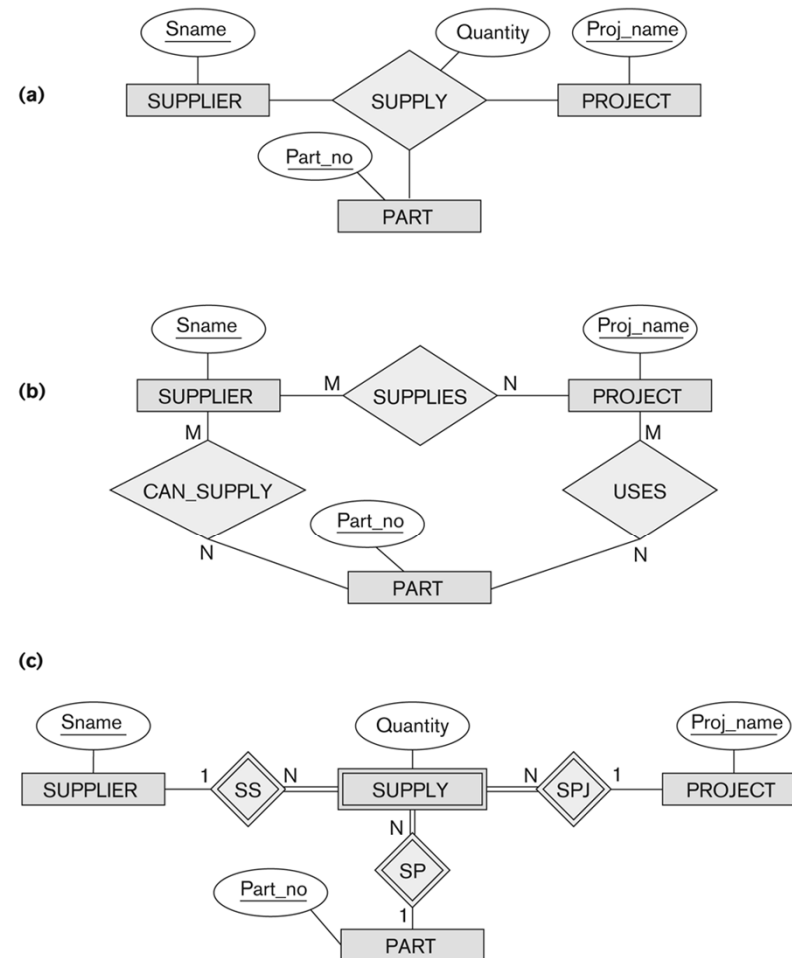


Figure 3.17

Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

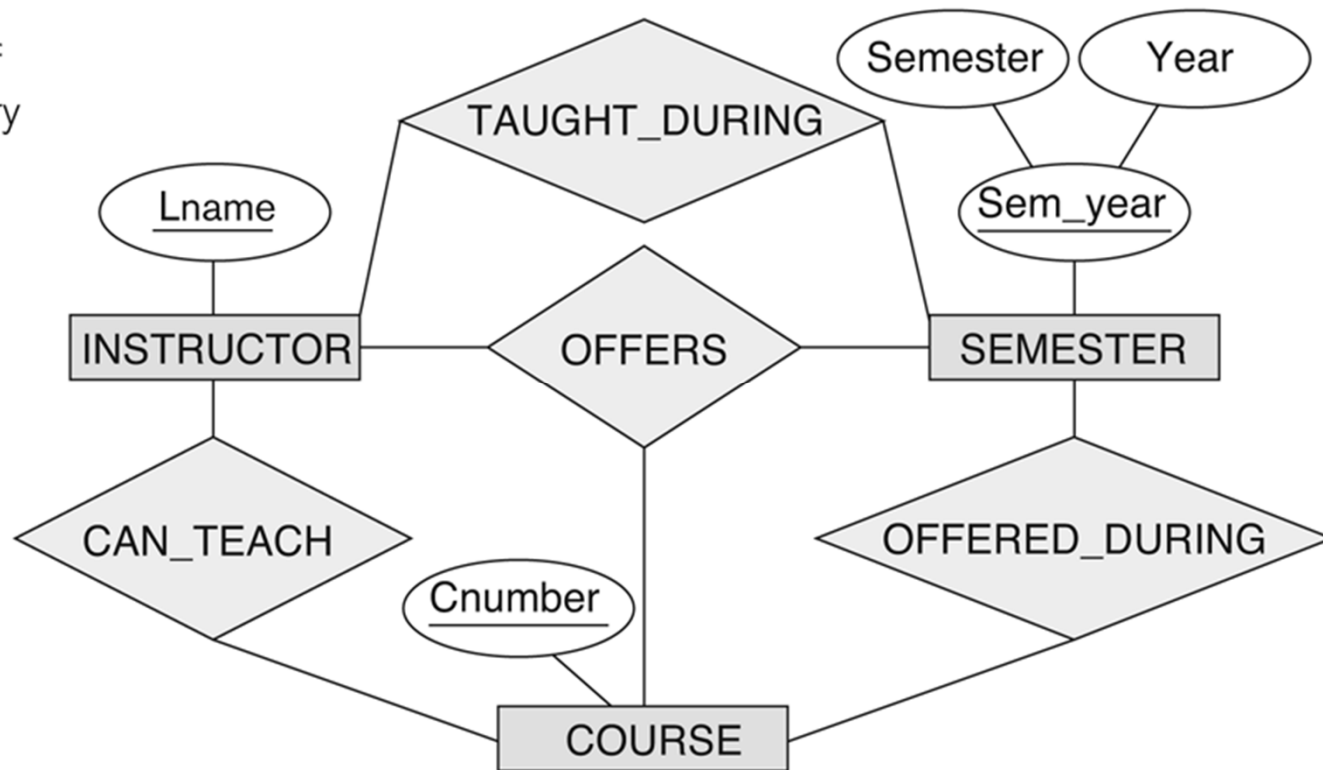
Discussion of n-ary relationships ($n > 2$)

- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant
- For example, the TAUGHT_DURING binary relationship in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)

Another example of a ternary relationship

Figure 3.18

Another example of ternary versus binary relationship types.



Displaying constraints on higher-degree relationships

- The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints
- Displaying a 1, M, or N indicates additional constraints
 - An M or N indicates no constraint
 - A 1 indicates that an entity can participate in at most one relationship instance *that has a particular combination of the other participating entities*
- In general, both (min, max) and 1, M, or N are needed to describe fully the constraints

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
- NEGATIVES:
 - Most tools lack a proper distinct notation for relationships with relationship attributes
 - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design

(See Chapter 12 for details)

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, space and security management
Oracle	Developer 2000/Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (Computer Associates)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++

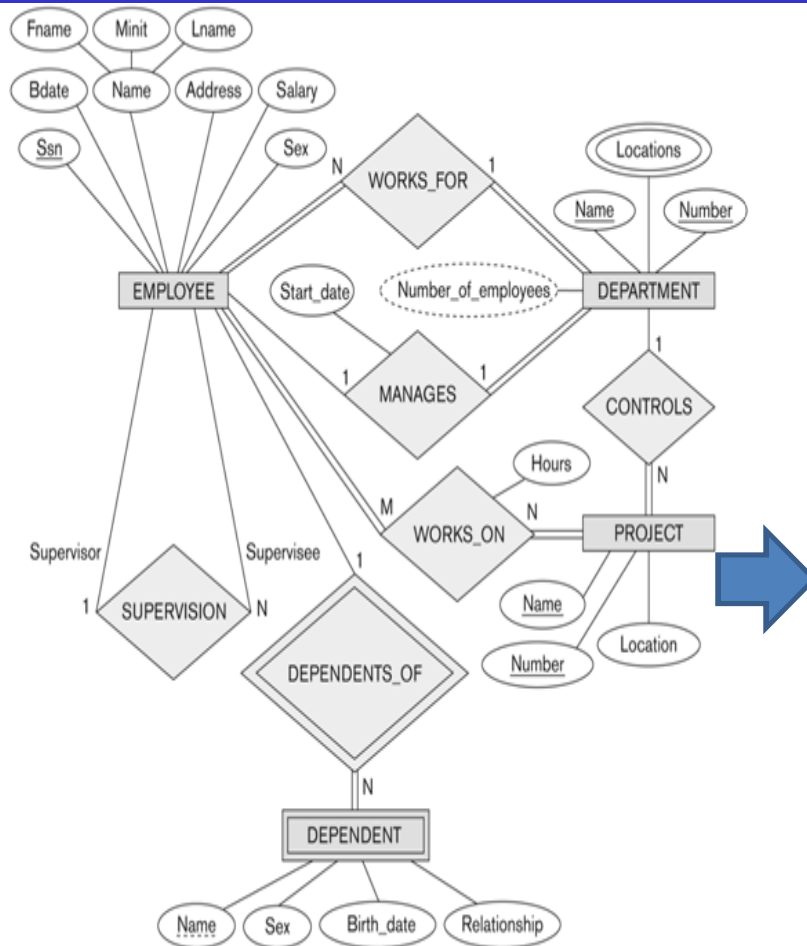
Extended Entity-Relationship (EER) Model (in next chapter)

- The entity relationship model in its original form did not support the specialization and generalization abstractions
- Next chapter illustrates how the ER model can be extended with
 - Type-subtype and set-subset relationships
 - Specialization/Generalization Hierarchies
 - Notation to display them in EER diagrams

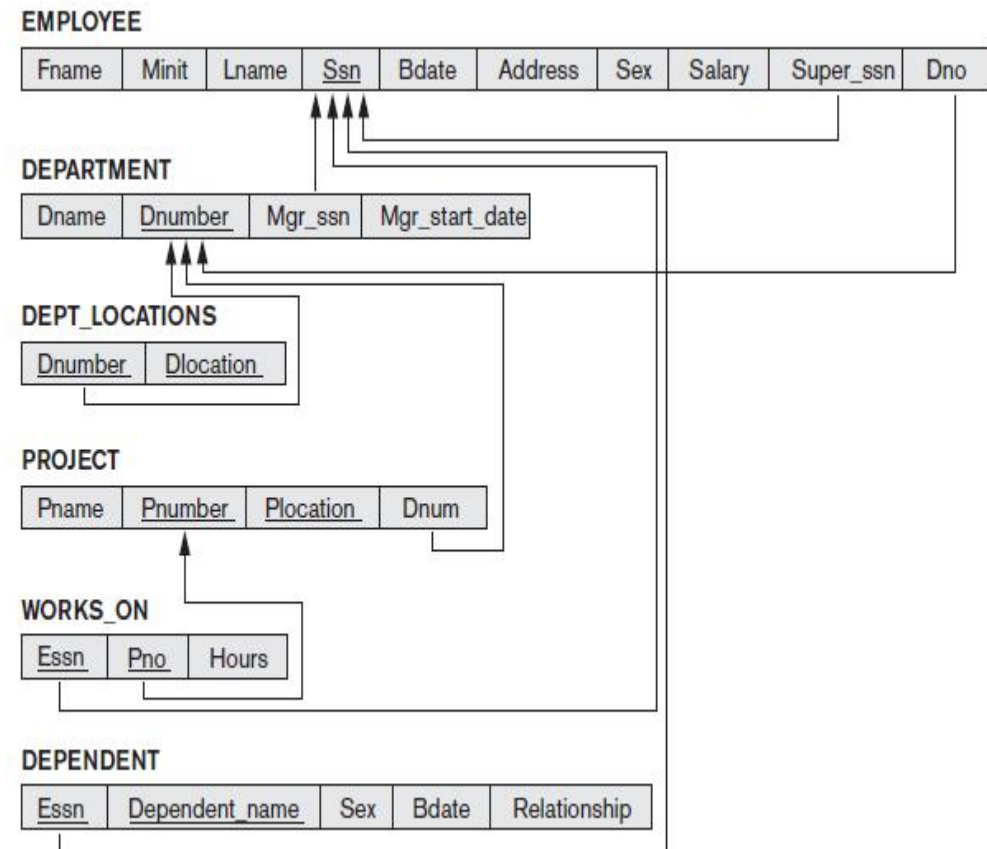
Chapter Summary

- ER Model Concepts: Entities, attributes, relationships
- Constraints in the ER model
- Using ER in step-by-step conceptual schema design for the COMPANY database
- ER Diagrams - Notation
- Alternative Notations – UML class diagrams, others

ER Model to Relational Mapping



Company ER Diagram

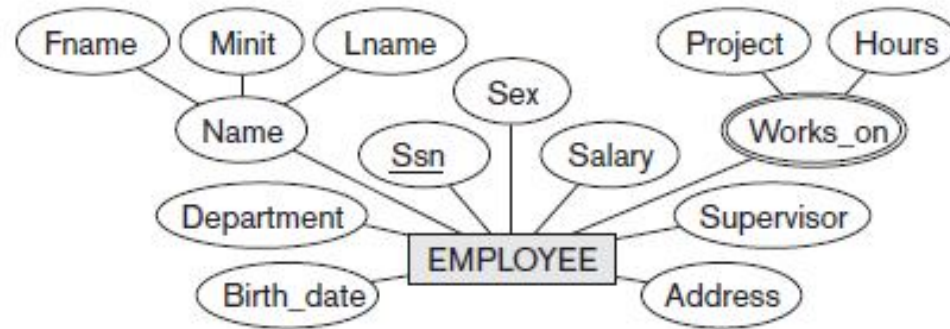


Company Relation Schema

Step 1: Converting Strong entity types

- Each entity type becomes a table
- Each single-valued attribute becomes a column
- Derived attributes are ignored
- Composite attributes are represented by components
- Multi-valued attributes are represented by a separate table
- The key attribute of the entity type becomes the primary key of the table

Strong Entity Type Mapping



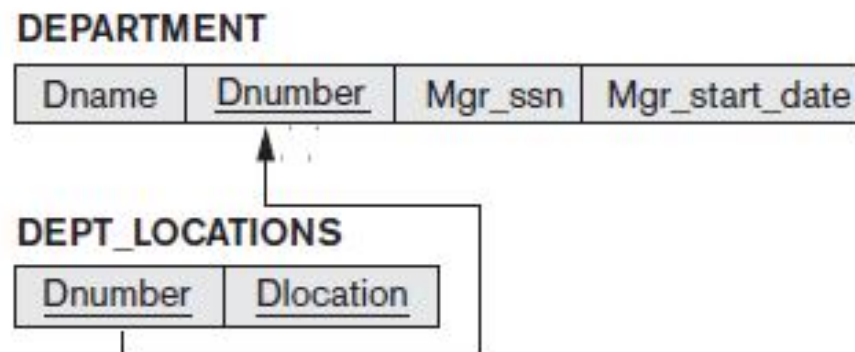
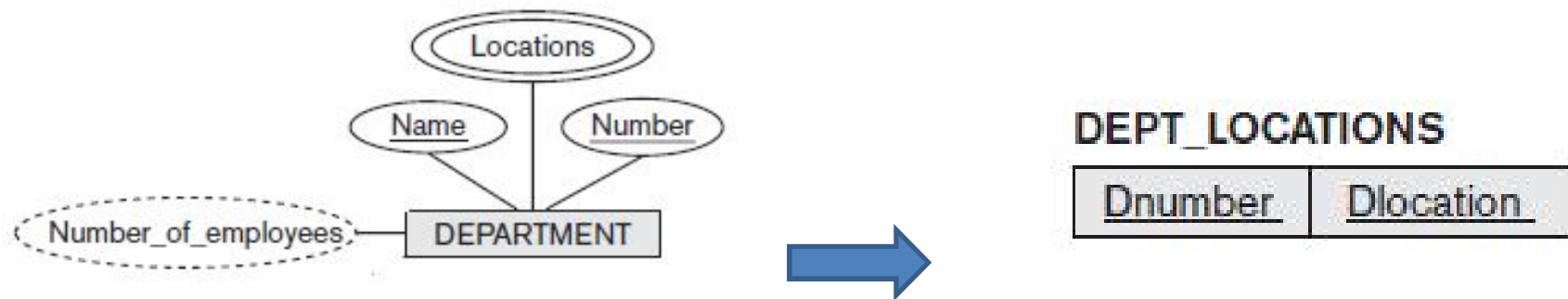
EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

Step2 : Multivalued Attribute

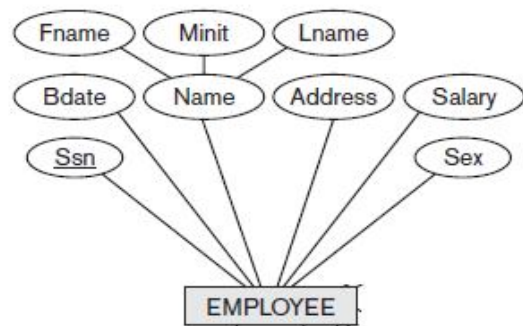
- For each multivalued attribute A , create a new relation R .
- This relation R will include an attribute corresponding to A , plus the primary key attribute K —as a foreign key in R —of the relation that represents the entity type or relationship type that has A as a multivalued attribute.
- The primary key of R is the combination of A and K . If the multivalued attribute is composite, include its simple components.

Multivalued Attribute



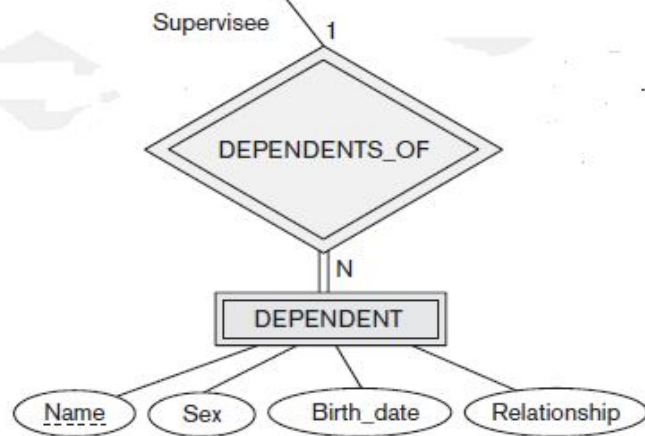
Step3: Mapping of Weak Entity Types

- For each weak entity type W with owner entity type E
 - create relation R including all simple attributes and simple component attributes of W
 - include primary key of E 's corresponding relation as a foreign key in W
 - primary key of R is combination of foreign key from E and W 's partial key



DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



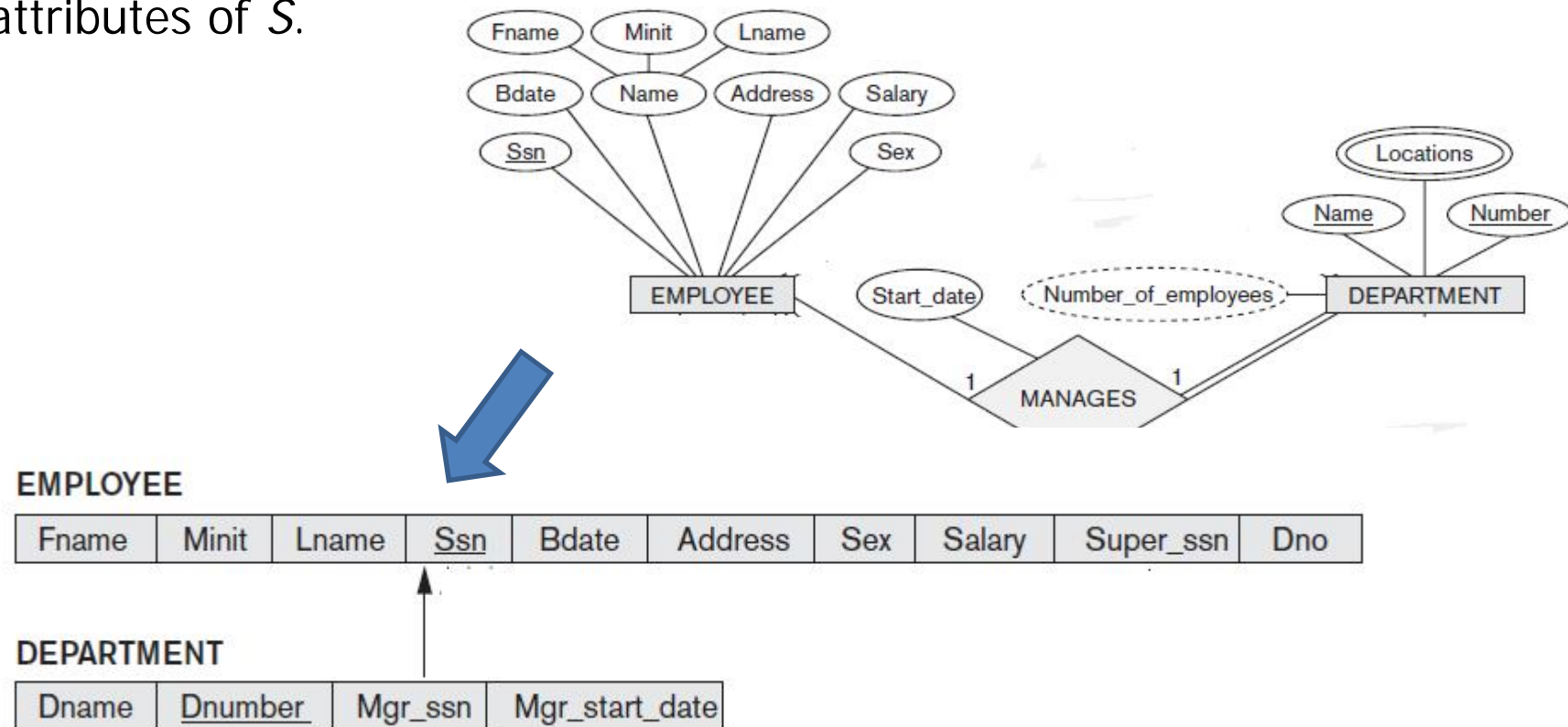
Step 4: Mapping of Binary 1:1 Relationship Types.

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R . There are three possible approaches:

- (1) Foreign key approach
- (2) Merged relationship approach
- (3) The cross-reference or relationship relation approach.

Foreign key approach

- Choose one of the relations— S , say—and include as a foreign key in S the primary key of T . It is better to choose an entity type with *total participation* in R in the role of S .
- Include all the simple attributes of the 1:1 relationship type R as attributes of S .



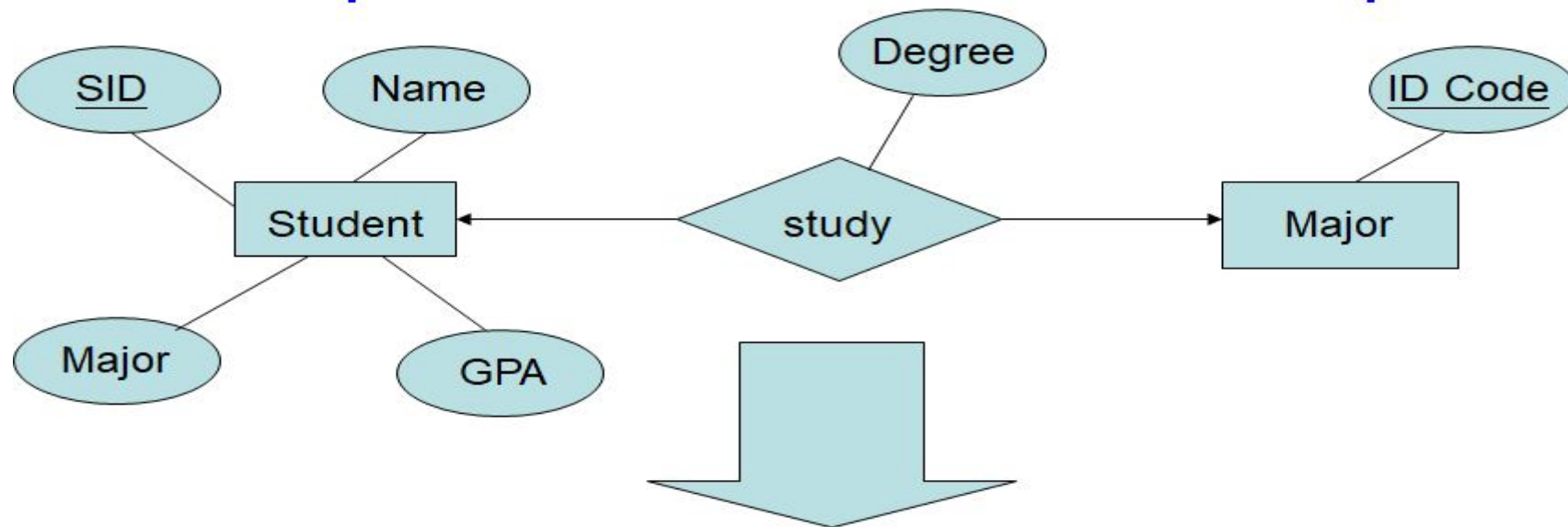
Merged relation approach:

- An alternative mapping of a 1:1 relationship
- type is to merge the two entity types and the relationship into a single relation.
- This is possible when *both participations are total*

Cross-reference or relationship relation approach.

- The third option is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types
- The relation R is called a **relationship relation**
- The relation R will include the primary key attributes of S and T as foreign keys to S and T . The primary key of R will be one of the two foreign keys, and the other foreign key will be a unique key of R .

Cross-reference(Example)

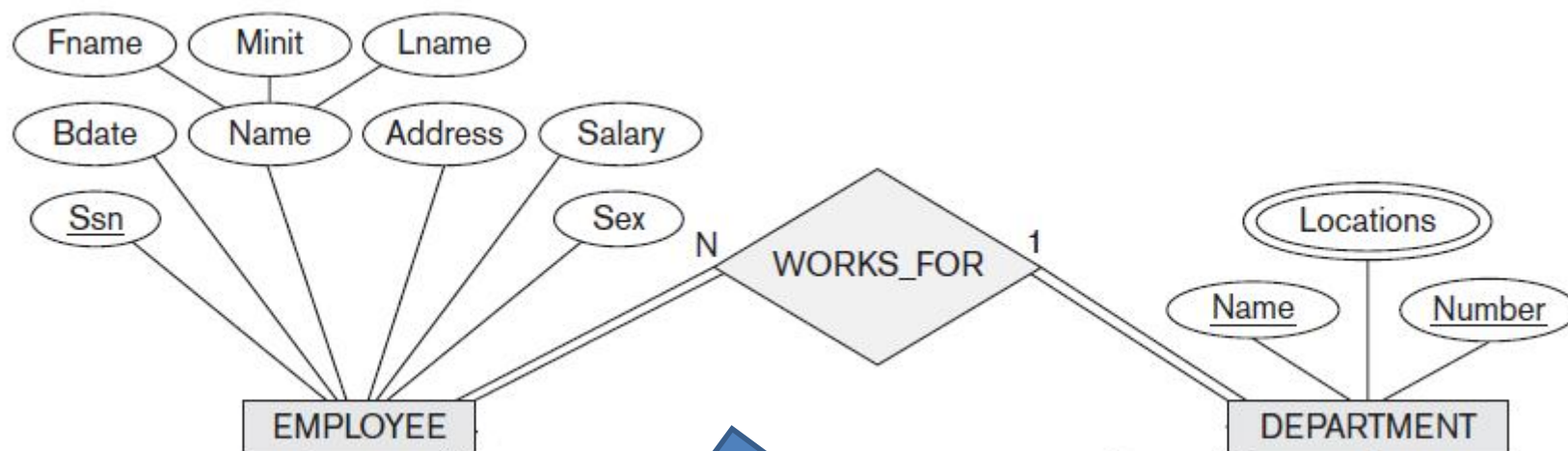


<u>SID</u>	<u>Maj ID Co</u>	S_Degree
9999	07	1234
8888	05	5678

Step 5: Mapping of Binary 1:N Relationship Types

- For each regular binary 1:N relationship type R , identify the relation S that represents the participating entity type at the *N-side* of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R
- In our example, map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION

Binary 1:N



EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

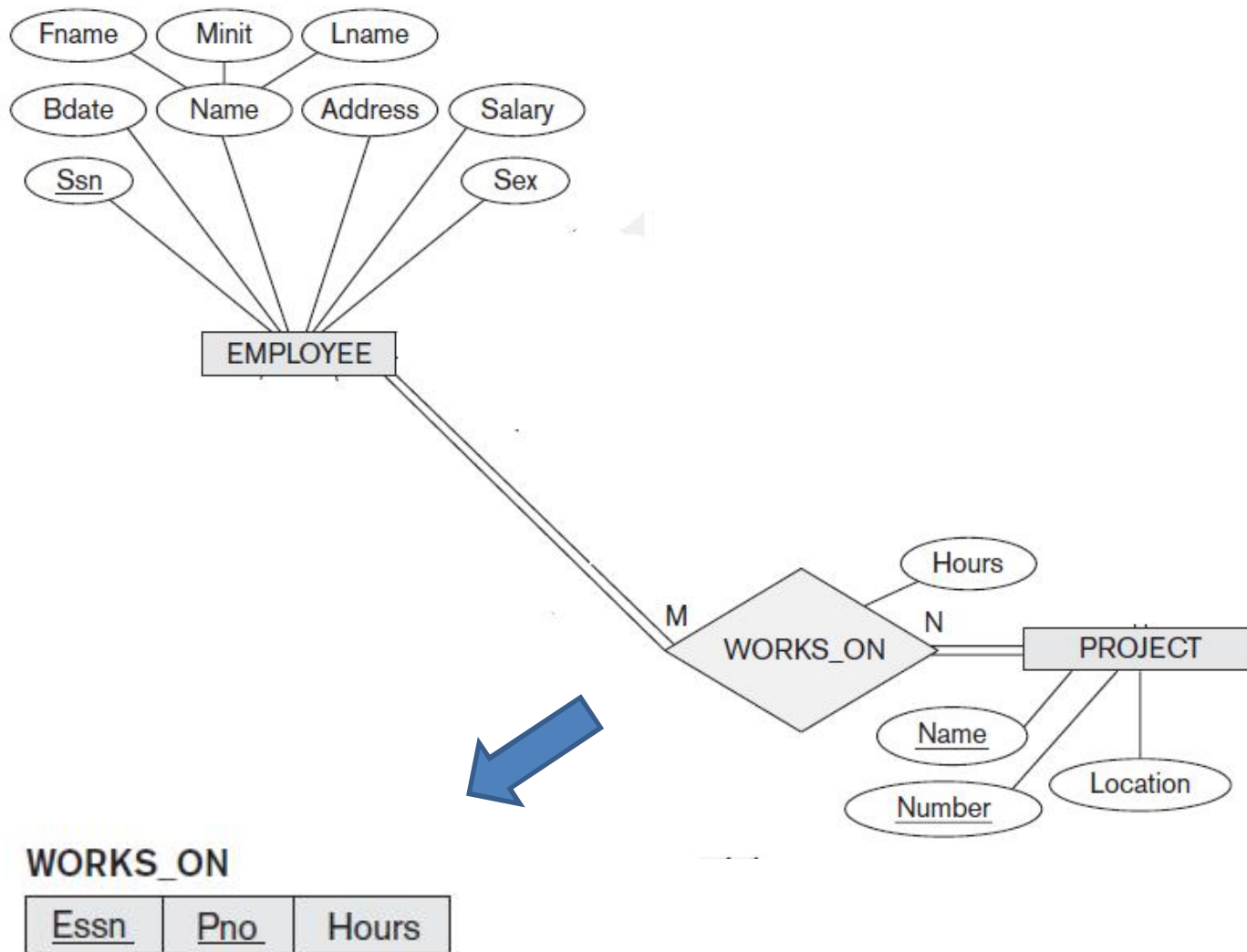
Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------



Step 6: Mapping of Binary M:N Relationship Types

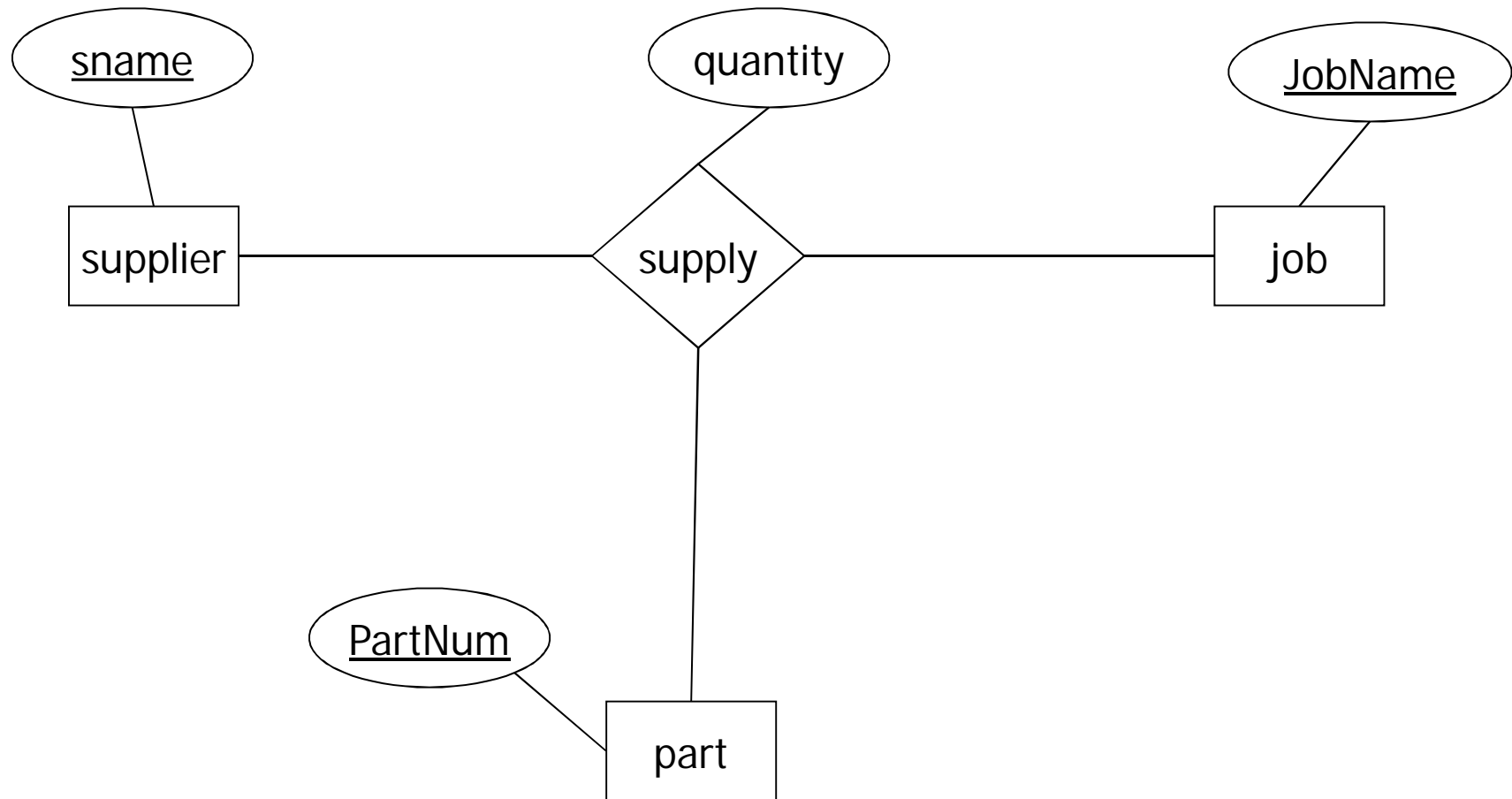
- Create a new relation S to represent R . Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their *combination* will form the primary key of S .
- Include any simple attributes of the M:N relationship type as attributes of S

Binary M:N



Step 7: Mapping of N -ary Relationship Types

- For each n -ary relationship type R , where $n > 2$, create a new relation S to represent R .
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the n -ary relationship type as attributes of S
- The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types





Tables that already exist:

Supplier (sname, ...)

Job (jname, ...)

Part (partNum, ...)

Add new Table:

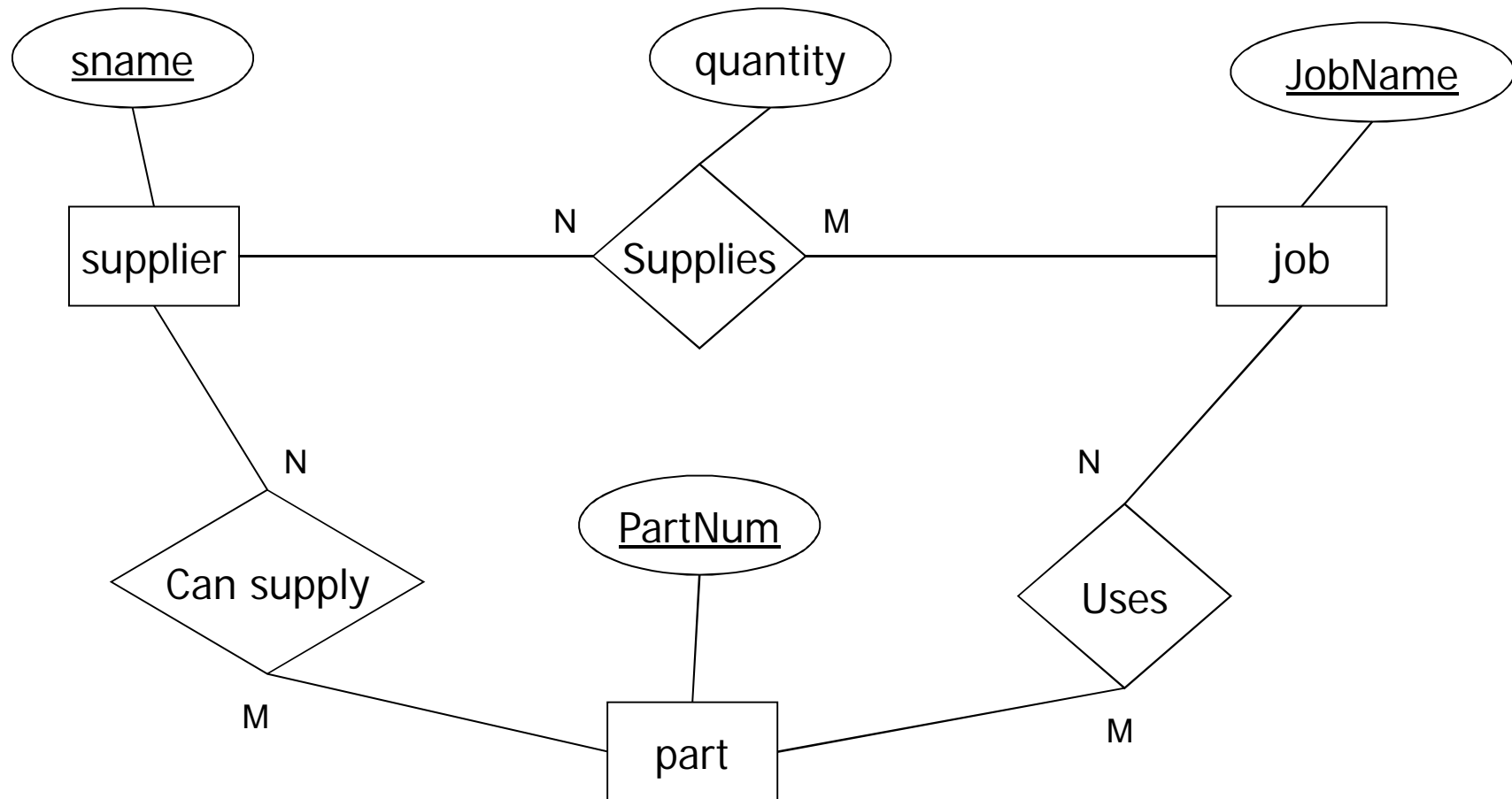
Supply (sname, jname, partNum, quantity, ...)

sname is FK -> Supplier (sname)

jname is FK -> Job (jname)

partNum is FK -> Part (partNum)

Binary version of n-ary example



Solution to binary version of n-ary

Supplier (sname, ...)

Job (jname, ...)

Part (partNum, ...)

Supplies (sname, jname)

Can_supply (sname, partNum)


Uses (partNum, jname)

Constraints

- **CREATE TABLE** emp_demo2 (employee_id
NUMBER(6) **PRIMARY KEY**, first_name
VARCHAR2(20) **NOT NULL**, last_name
VARCHAR2(25) **NOT NULL**, email
VARCHAR2(25) **UNIQUE**, phone_number
VARCHAR2(20) **UNIQUE**, job_id
VARCHAR2(10), salary NUMBER(8,2)
CHECK(SALARY>0), deptid NUMBER(4)
) ;

Constraints with Name

```
CREATE TABLE emp_demo3  
( employee_id    NUMBER(6) CONSTRAINT emp_eid PRIMARY  
KEY,  
  first_name     VARCHAR2(20),  
  last_name      VARCHAR2(25) CONSTRAINT emp_last_name_nn  
NOT NULL,  
  email          VARCHAR2(25) CONSTRAINT emp_email_nn NOT  
NULL,  
  phone_number   VARCHAR2(20),  job_id      VARCHAR2(10)  
CONSTRAINT emp_job_nn NOT NULL,  salary  
  NUMBER(8,2) CONSTRAINT emp_salary_nn NOT NULL,  
  deptid         NUMBER(4), CONSTRAINT emp_dept FOREIGN  
KEY(deptid) REFERENCES department(deptid) ,  
CONSTRAINT emp_salary_min CHECK (salary > 0) ,  
CONSTRAINT emp_email_uk UNIQUE (email) ) ;
```



```
CREATE TABLE emp_demo4  ( emp_id  
NUMBER(6),      emp_name  VARCHAR2(15),  
      salary    NUMBER(10) CHECK (salary between  
1000 and 10000)  
      );
```

Adding Constraints

ALTER TABLE <tablename> ADD CONSTRAINT <constraint_name> constraint_type (<column name>);

Examples:

ALTER TABLE emp_demo4 ADD CONSTRAINT con_pk1 PRIMARY KEY(emp_id);

ALTER TABLE emp_demo4 ADD CONSTRAINT con_emp_uk UNIQUE(phoneno);

ALTER TABLE emp_demo4 ADD CONSTRAINT con_empfk FOREIGN KEY(DNO) REFERENCES department(dno);

ALTER TABLE emp_demo4 ADD CONSTRAINT con_emp_ck CHECK (salary >0);

ALTER TABLE emp_demo4 MODIFY (<Column name> <datatype> CONSTRAINT constraint_name NOT NULL);

Drop Constraints

- **ALTER TABLE <tablename> DROP CONSTRAINT < constraint name**
- **ALTER TABLE employees DROP UNIQUE (email);**

ON DELETE CASCADE

A foreign key with a cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete.

```
CREATE TABLE supplier  
(supplier_id number(10)not null,  
supplier_name varchar2(50)not null,  
contact_name varchar2(50),  
CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));
```

```
CREATE TABLE products  
(product_id number(10)not null,  
suppl_id number(10) not null,  
CONSTRAINT fk_supplier FOREIGN KEY (suppl_id) REFERENCES  
supplier(supplier_id) ON DELETE CASCADE);
```

Because of the cascade delete, when a record with a particular supplier_id is deleted from supplier table, then all the records of the same supplier_id will be deleted from products table also.

On Delete set null/On Delete Set Default

- A **foreign key** with "**set null on delete**" means that if a record in the parent table is **deleted**, then the corresponding records in the child table will have the **foreign key** fields **set to null**. The records in the child table will not be **deleted**.
- create table Users (UserID int primary key, UserName varchar(100), ThemeID int default 1 constraint Users_ThemeID_FK references Themes(ThemeID) on delete set default)