

The background of the slide features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the left and right sides, framing the central text area.

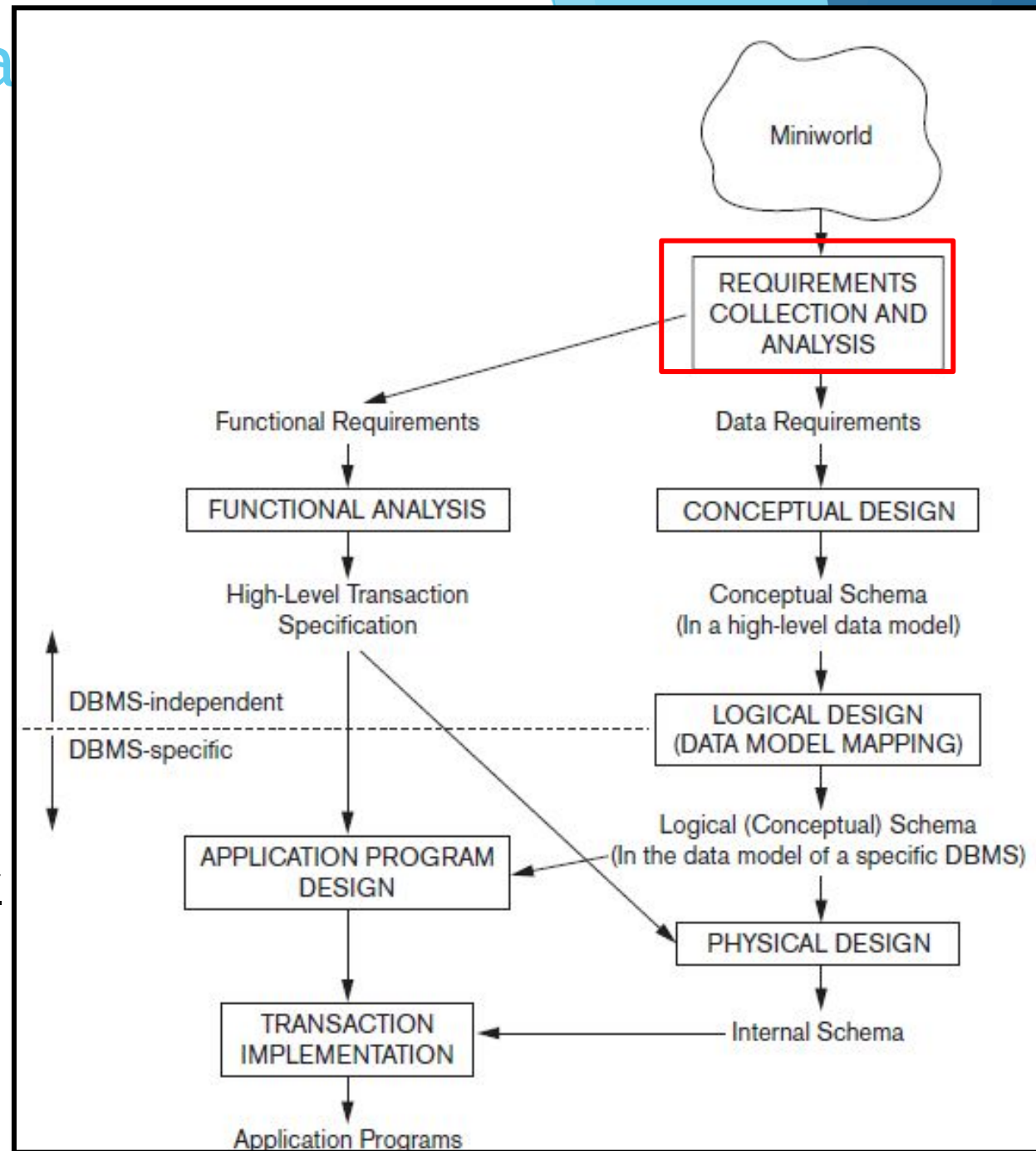
Chapter 3 - Data Modeling Using the Entity- Relationship (ER) Model

Content

- Using High-Level Conceptual Data Models for Database Design
- A Sample Database Application
- Entity Types, Entity Sets, Attributes and Keys
- Relationship Types, Relationship Sets, Roles, and Structural Constraints
- Weak Entity Types
- Refining the ER Design for the COMPANY Database
- ER Diagrams, Naming Conventions, and Design Issues
- Relationship Types of Degree Higher than Two

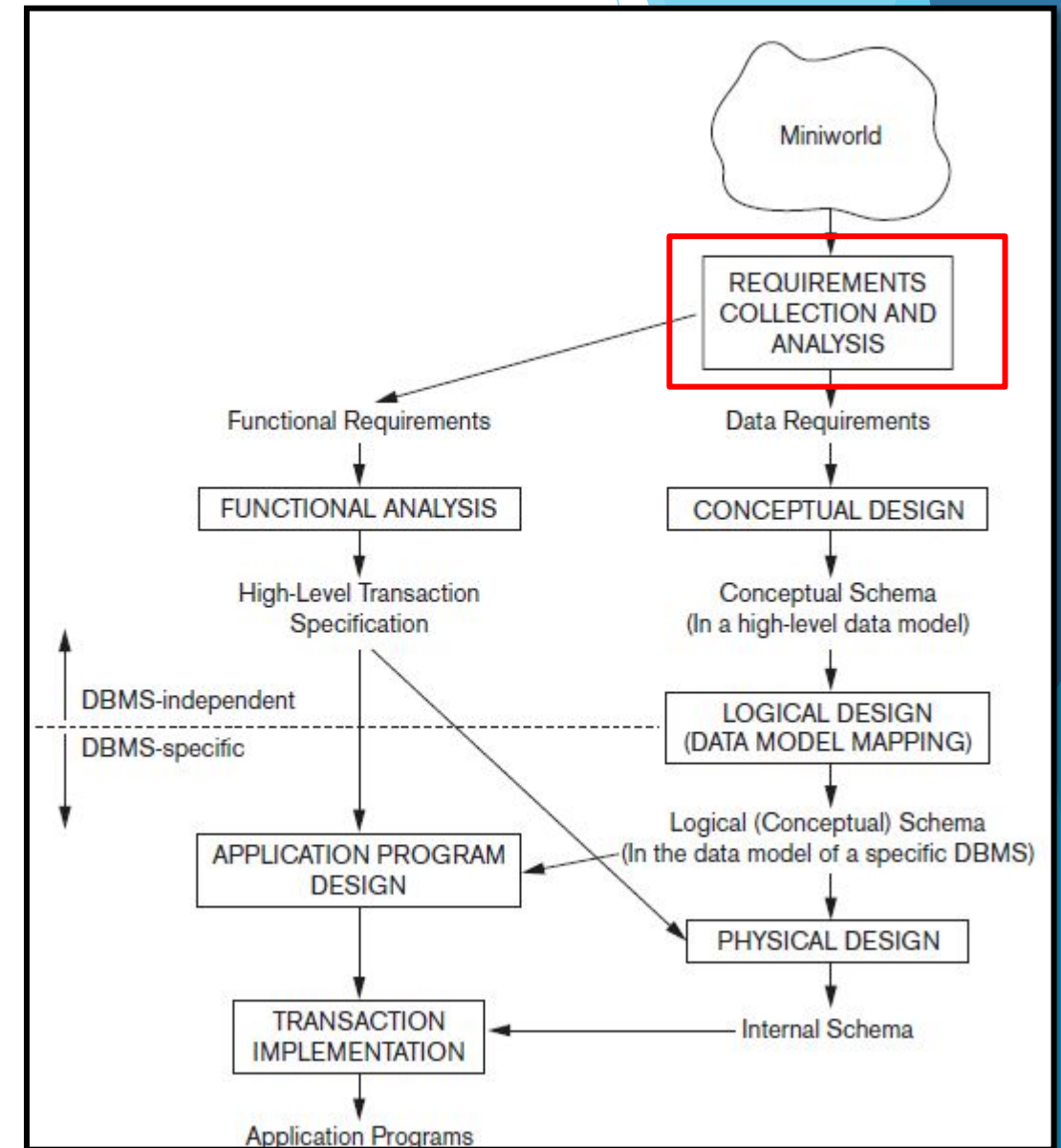
Using High-Level Conceptual Data Models for Database Design

- Overview of the database design process.
- STEP 1: **Requirements collection and analysis.**
- The DB designers interview DB users to understand and document their **data requirements**.
- The **result** of this step is a concisely written set of **users' requirements**.
 - a description of the data used;
 - the details of how data is to be used;
 - any additional requirements for the new database system(security etc.).



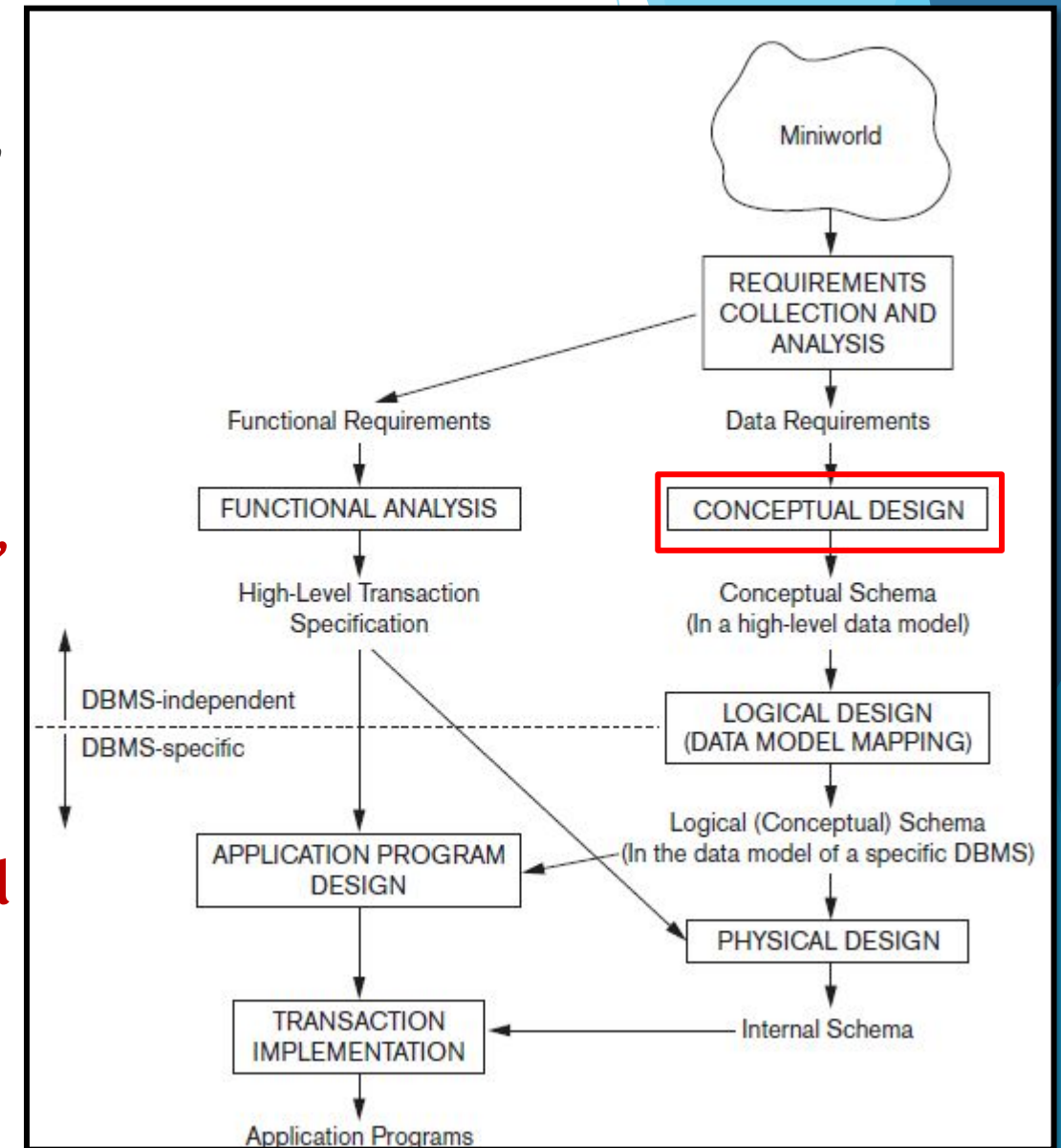
Using High-Level Conceptual Data Models for Database Design

- ▶ with listing the data requirements, it is useful to **specify the known functional requirements of the application.**
 - ▶ Functional requirements: User defined operations applied to the database, including both retrievals and updates.
- ▶ For functional requirements specifications, **data flow diagrams, sequence diagrams, scenarios**, and other techniques are used.



Using High-Level Conceptual Data Models for Database Design

- ▶ **STEP 2: conceptual design**
 - ▶ create a **conceptual schema** for the database, using a high-level conceptual data model.
 - ▶ conceptual schema:
 - ▶ concise description of the **user data requirements**
 - ▶ And detailed descriptions of the **entity types, relationships, and constraints**.
- ▶ No implementation details included
- ▶ **easier to understand**
- ▶ **Useful to communicate with nontechnical users.**



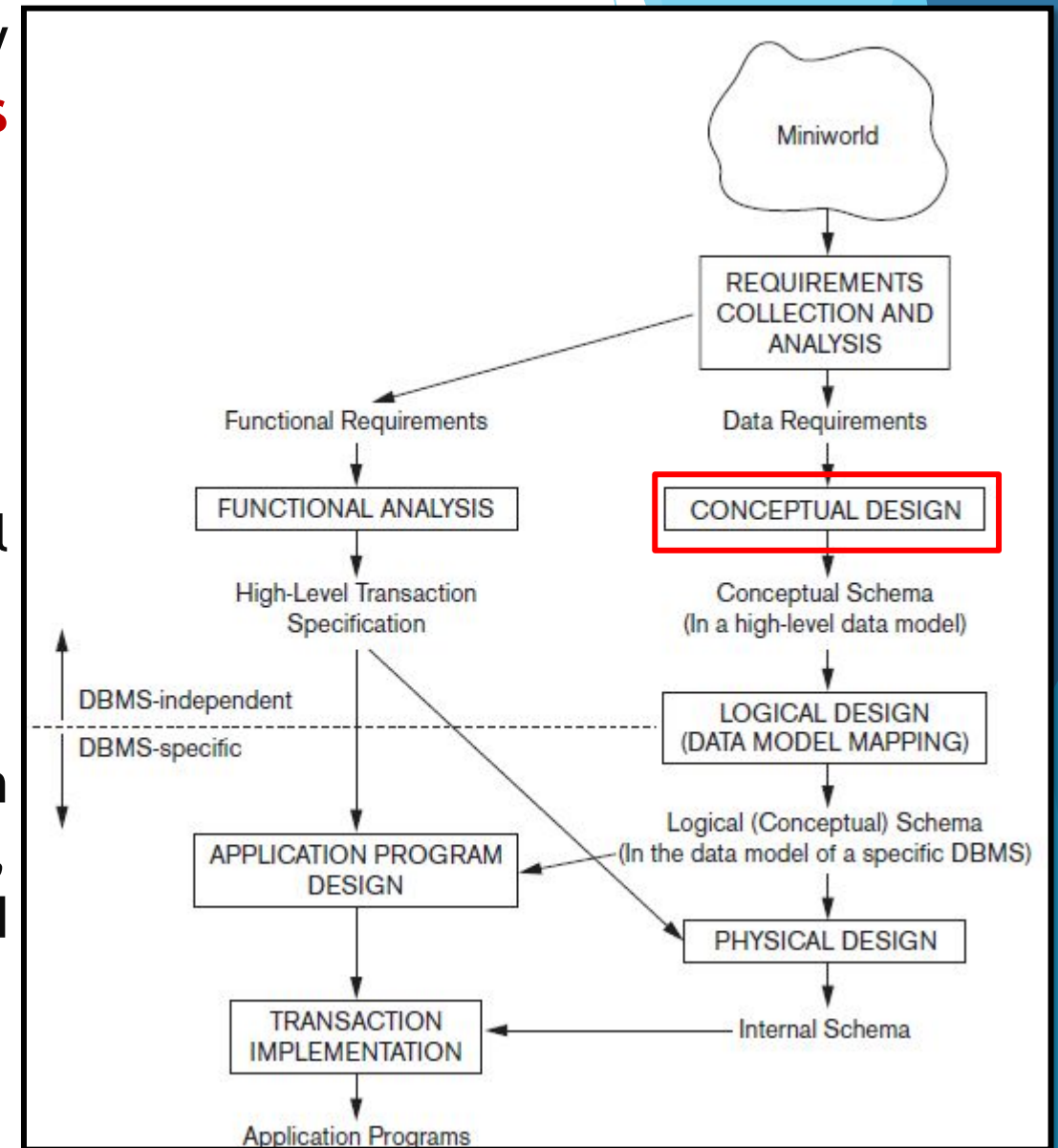
Using High-Level Conceptual Data Models for Database Design

▶ Conceptual database design is entirely **independent of implementation details** such as

- ▶ the target DBMS software,
- ▶ application programs,
- ▶ programming languages,
- ▶ hardware platform, or any other physical considerations.

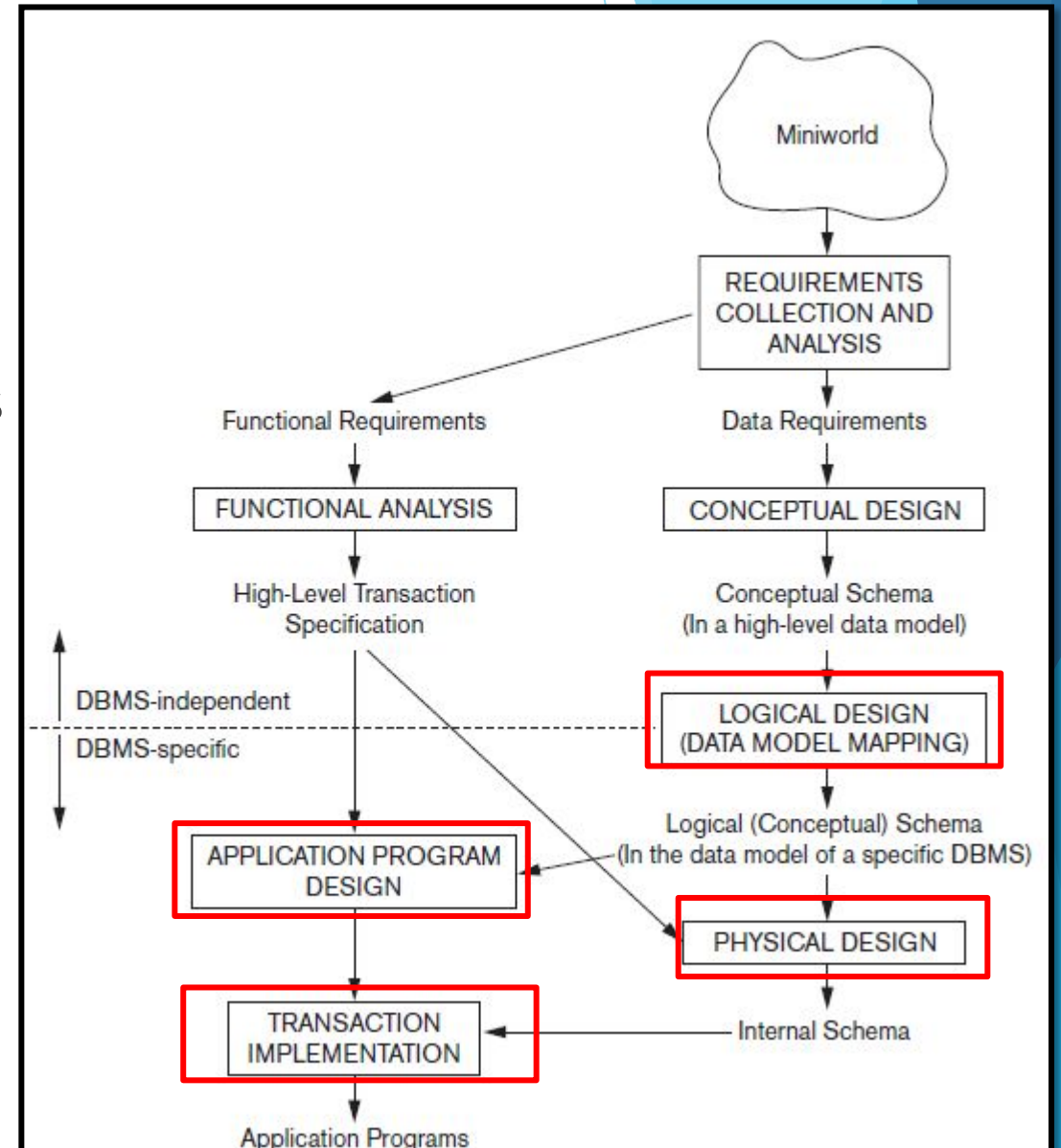
▶ Advantage: **a good database design**

- ▶ AS database designers can focus on specifying the properties of the data, without thinking about storage and implementation details.



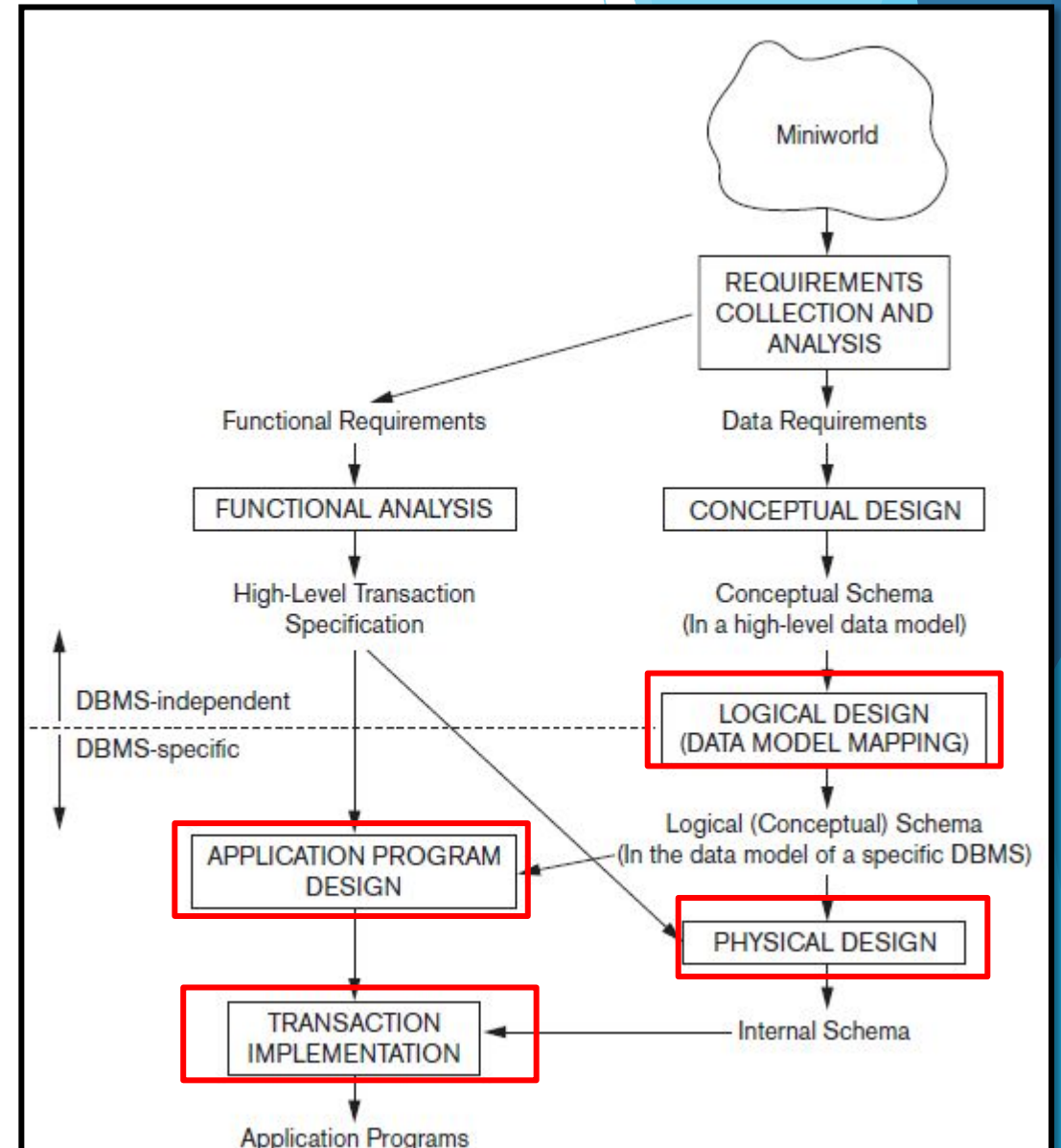
Using High-Level Conceptual Data Models for Database Design

- ▶ STEP 3: logical design/data model mapping
 - ▶ the actual implementation of the database, using a commercial DBMS.
 - ▶ Use of an implementation data model—such as the relational (SQL) model
 - ▶ the conceptual schema transformation:
 - ▶ high-level data model -> implementation data model.
- ▶ **Result: database schema in the implementation data model of the DBMS.**



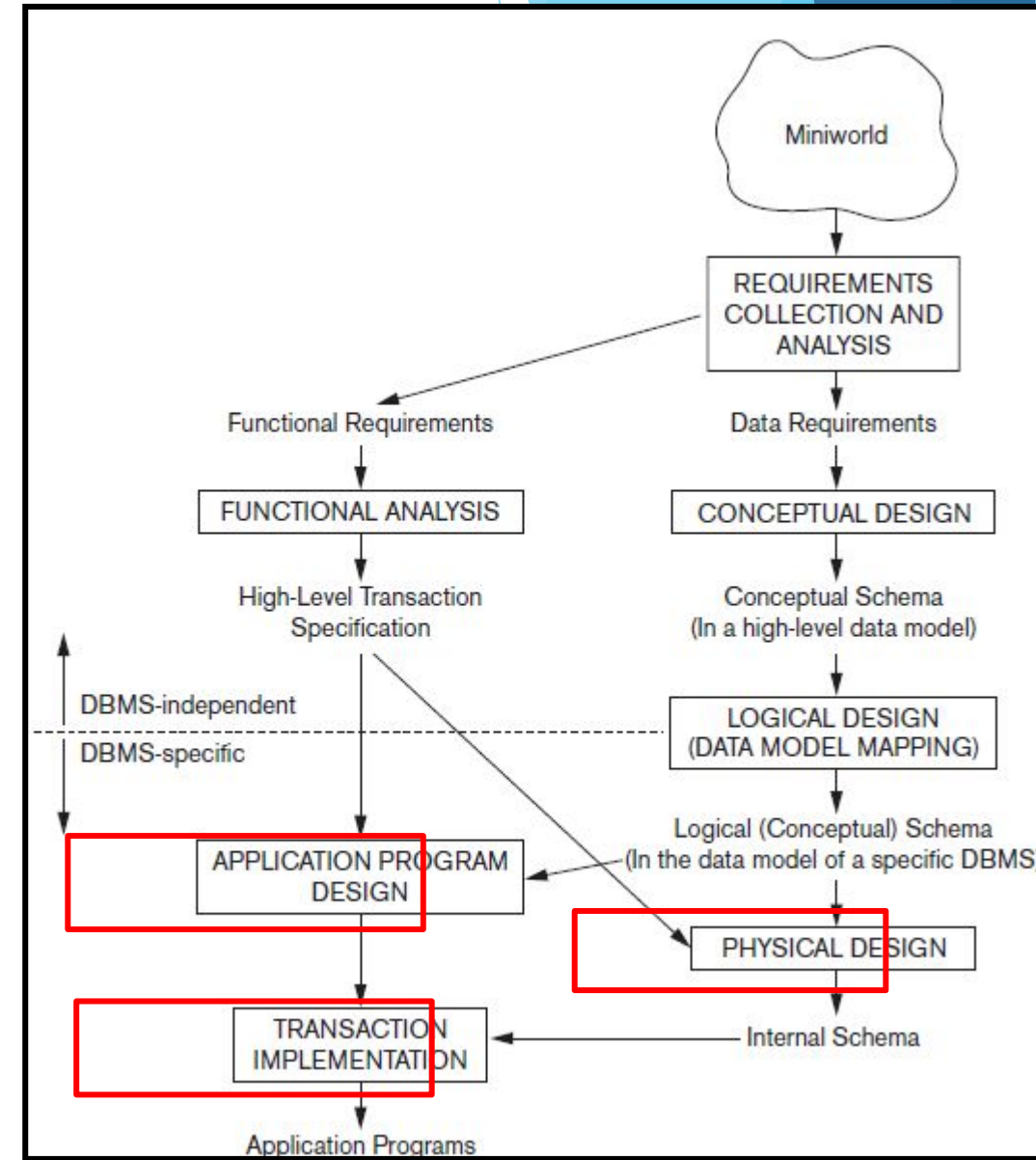
Using High-Level Conceptual Data Models for Database Design

- ▶ A logical model is derived depending upon the underlying data model of the target DBMS.
 - ▶ In other words, we know that the DBMS is, for example, relational or object-oriented.
- ▶ Here other aspects of the chosen DBMS like physical details, such as storage structures or indexes are ignored.



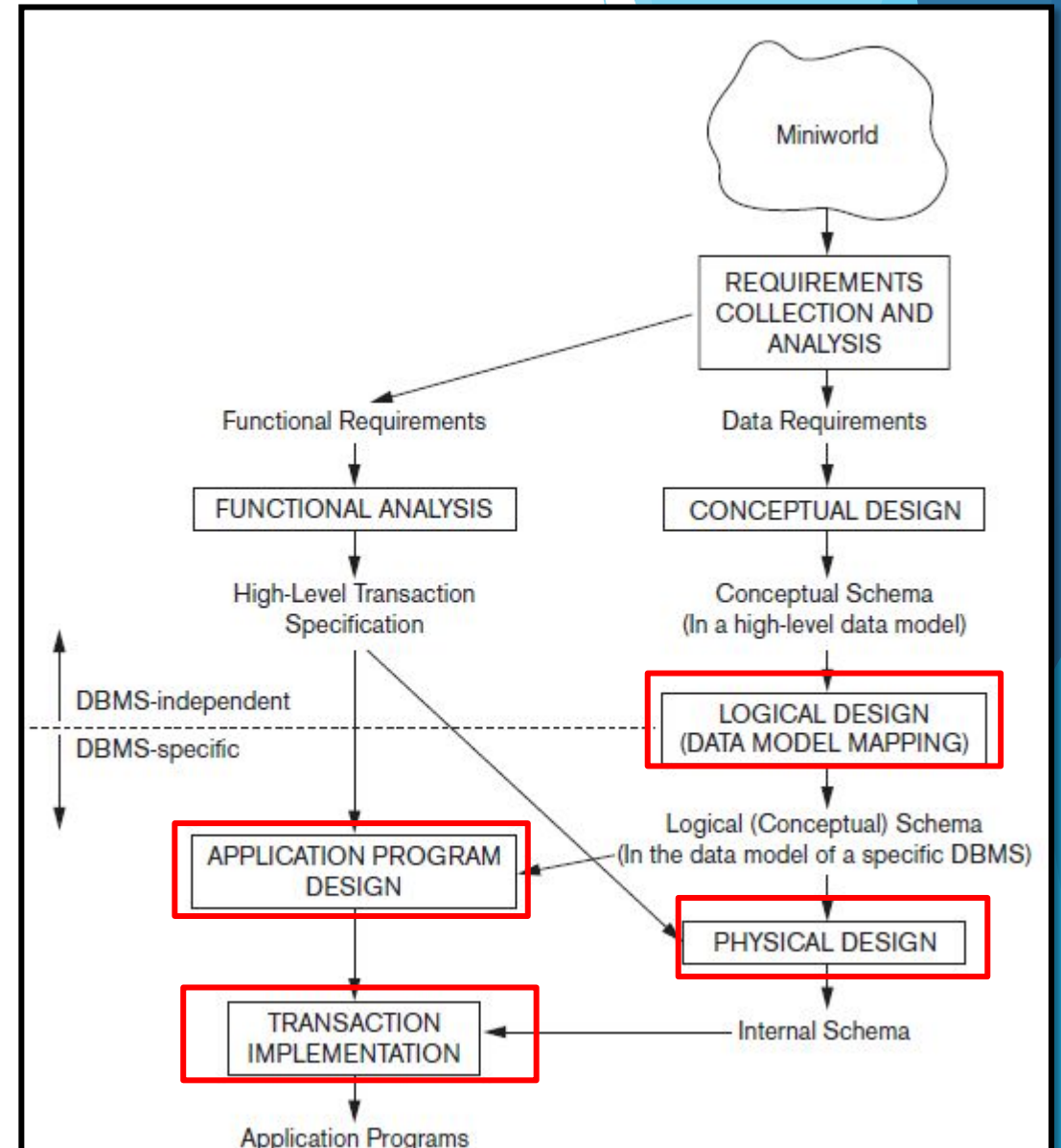
Using High-Level Conceptual Data Models for Database Design

- STEP4: **physical design phase**
- describe how we intend to physically implement the logical database design.
- For the relational model, this involves:
 - creating a set of relational tables and the constraints on these tables derived from the information presented in the logical data model.
 - identifying the storage structures and access methods for the data to achieve an optimum performance for the database system.



Using High-Level Conceptual Data Models for Database Design

- ▶ In parallel with these activities,
 - ▶ **application programs** are designed
 - ▶ and implemented as **database transactions** corresponding to the high-level transaction specifications.



A Sample Database Application

- ▶ We describe a sample database application, called COMPANY, which serves to illustrate the basic ER model concepts and their use in schema design.
- ▶ We list the data requirements for the database here, and then create its conceptual schema step-by-step as we introduce the modeling concepts of the ER model.
- ▶ The COMPANY database keeps track of a company's employees, departments, and projects.

Entity Types, Entity Sets, Attributes, and Keys

Entities and Attributes

Entity: is a thing or object in the real world with an independent existence.

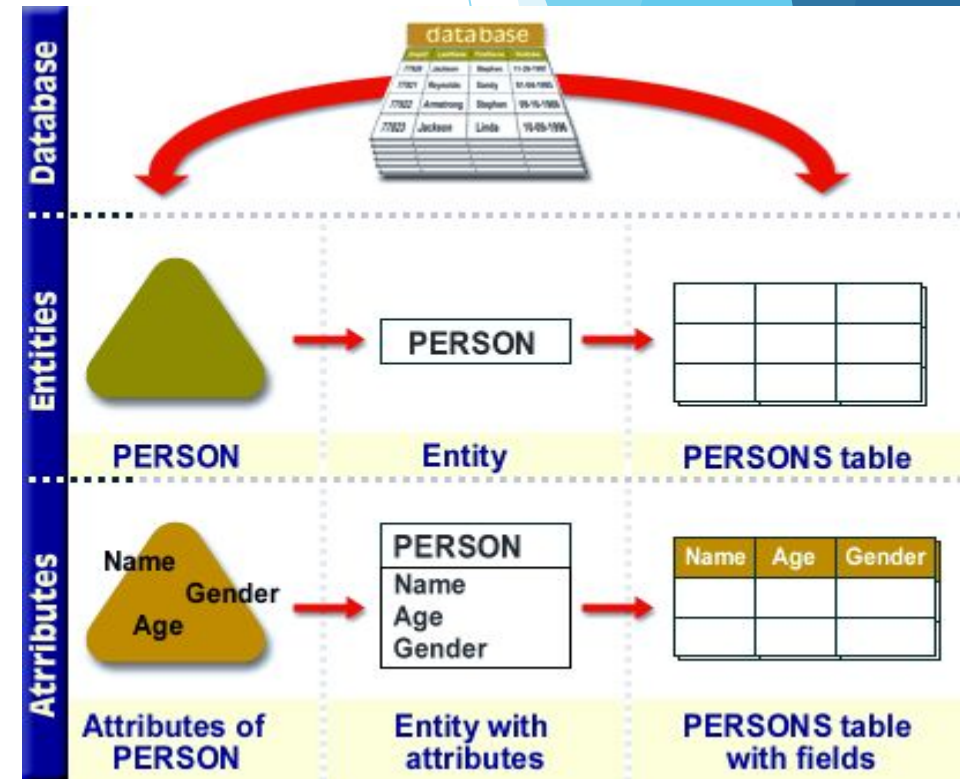
- An entity may be an object with a physical existence (for example, a particular **person**, **car**, **house**, or **employee**)

or

- it may be an object with a conceptual existence (for instance, **a job**, or **a university course**).

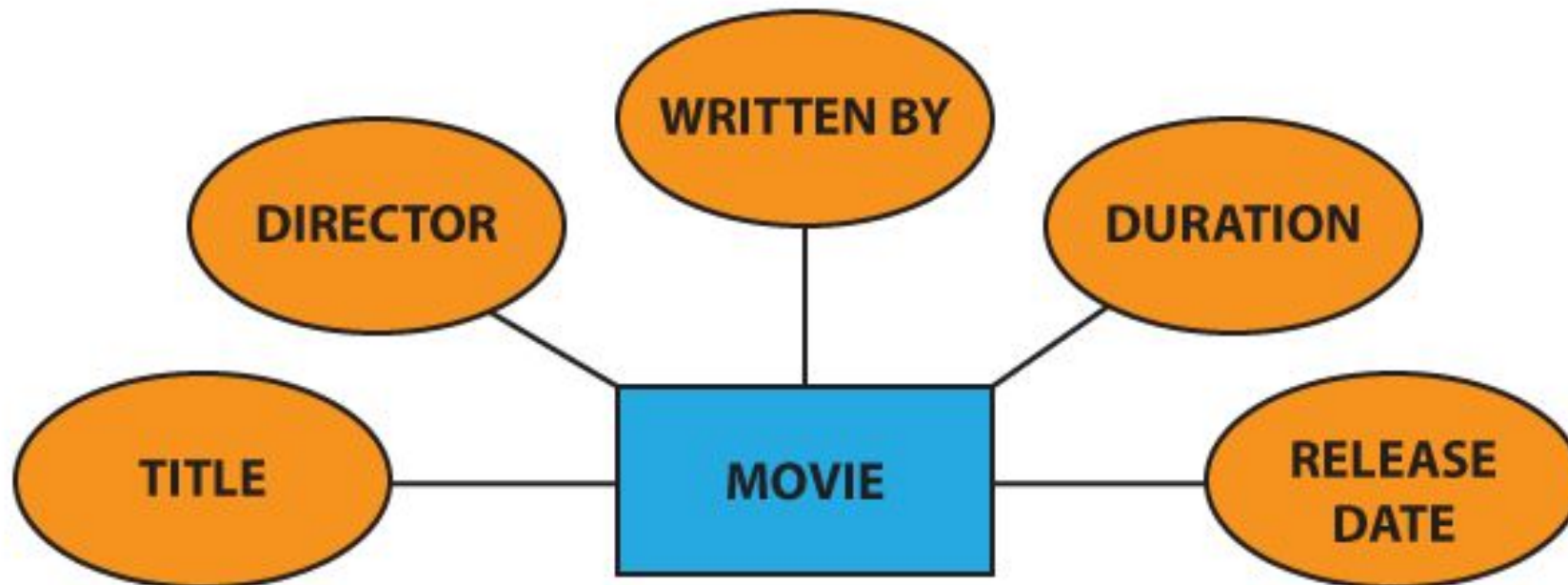
Attributes: properties that describe an entity.

- For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Representation in ER



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ The EMPLOYEE entity e_1 has four attributes: Name, Address, Age, and Home_phone; their values are 'John Smith,' '2311 Kirby, Houston, Texas 77001', '55', and '713-749-2630', respectively.
- ▶ The COMPANY entity c_1 has three attributes: Name, Headquarters, and President; their values are 'Sunco Oil', 'Houston', and 'John Smith', respectively.
- ▶ types of attributes: simple/composite, single valued/multivalued, and stored/derived.

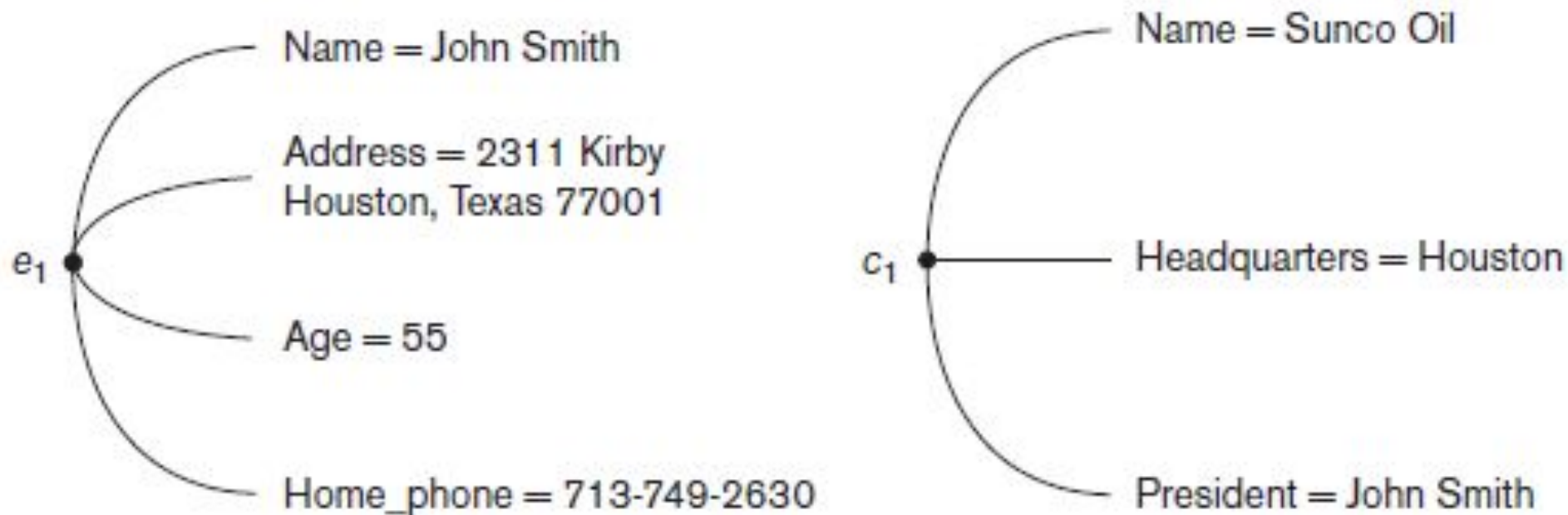
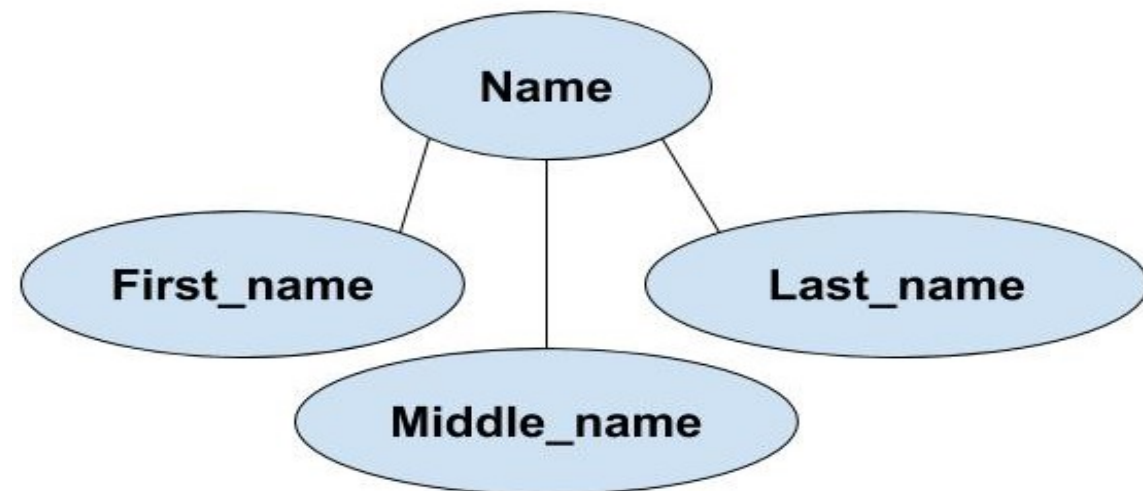
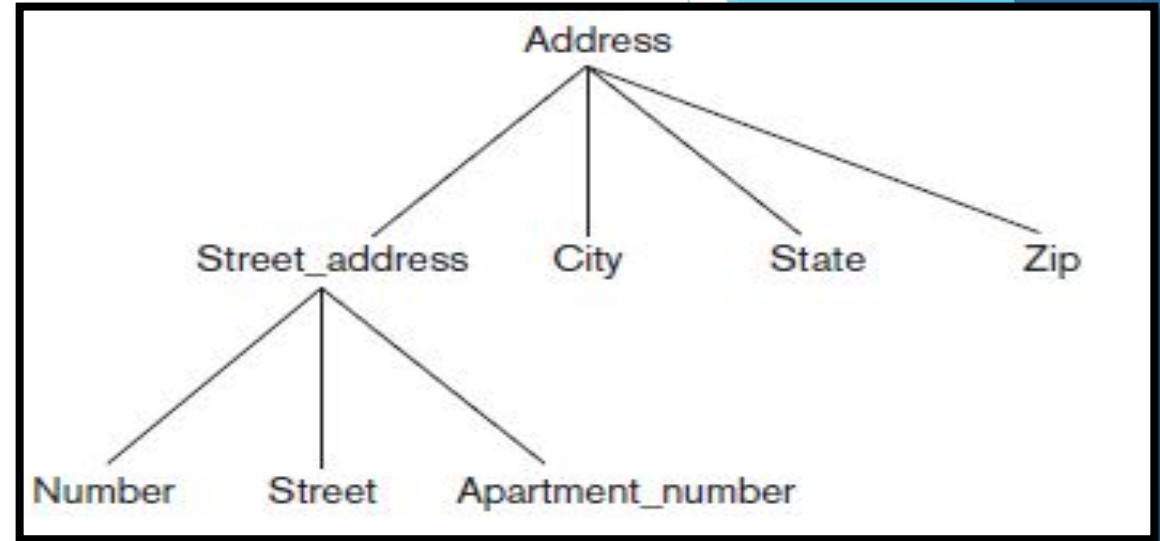


Figure 3.3
Two entities,
EMPLOYEE e_1 , and
COMPANY c_1 , and
their attributes.

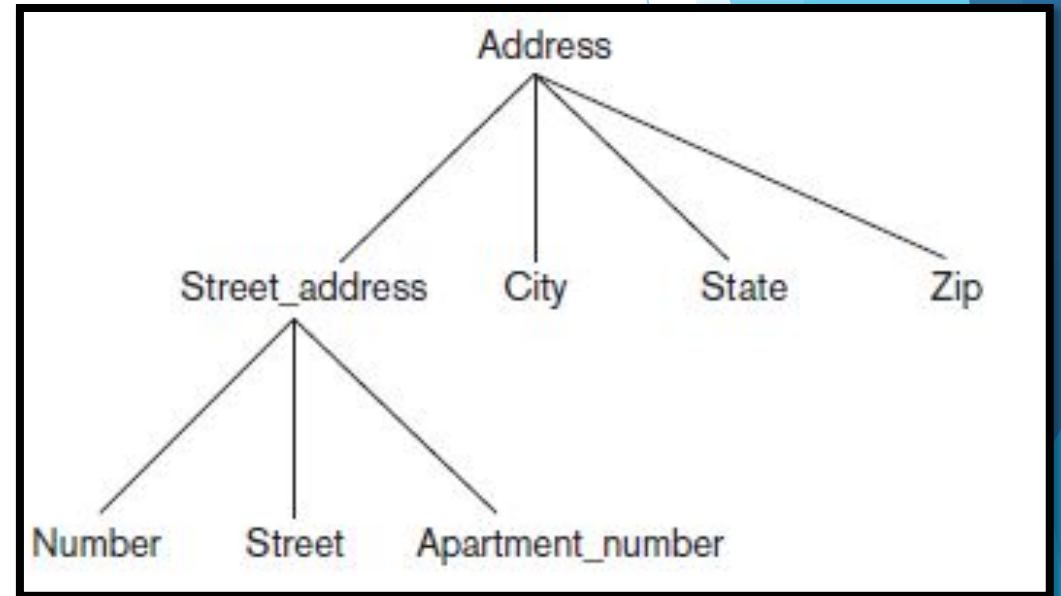
Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Composite versus Simple (Atomic) Attributes
- ▶ **Composite attributes**
 - ▶ can be divided into smaller subparts, which represent more basic attributes with independent meanings.
 - ▶ Composite attributes can form a hierarchy:
 - ▶ for example, Street_address can be further subdivided into three simple component attributes: Number, Street, and Apartment_number, as shown in Figure 3.4.
- ▶ The value of a composite attribute = concatenation of the values of its component simple attributes.



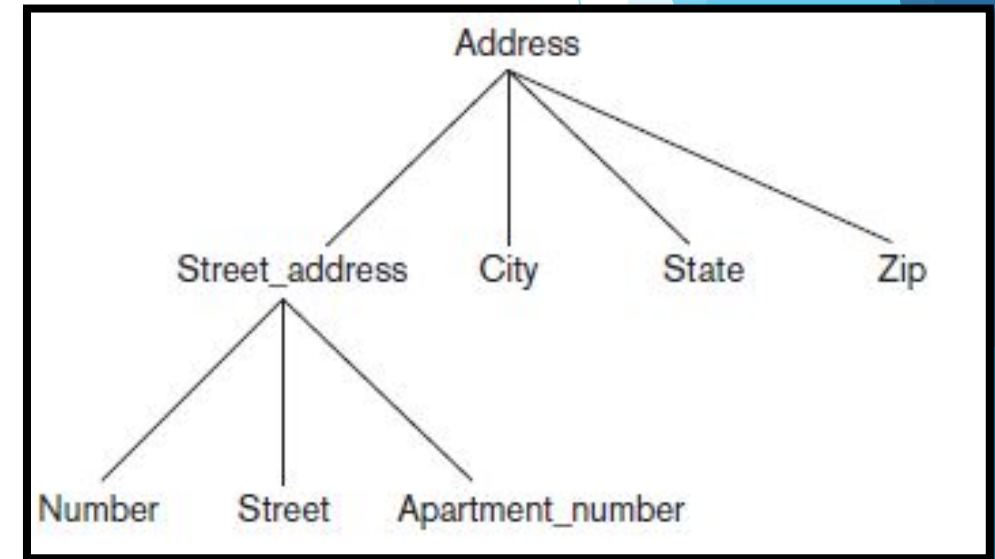
Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Composite versus Simple (Atomic) Attributes
- ▶ For example, the Address attribute of the EMPLOYEE entity shown in Figure 3.3 can be subdivided into Street_address, City, State, and Zip, with the values '2311 Kirby', 'Houston', 'Texas', and '77001'.
- ▶ **simple or atomic attributes:** Attributes that are not divisible



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Composite versus Simple (Atomic) Attributes
- ▶ Where are Composite attributes used?
 - ▶ Useful to model situations in which a user sometimes refers to the composite attribute as a unit but at other times refers specifically to its components.
- ▶ If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.
 - ▶ For example, if there is no need to refer to the individual components of an address (Zip Code, street, and so on), then the whole address can be designated as a simple attribute.



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Single-Valued versus Multivalued Attributes.
- ▶ **Single-valued:** Attributes have a single value for a particular entity.
 - ▶ For example, Age is a single-valued attribute of a person.
- ▶ An attribute can have a set of values for the same entity
 - ▶ a Colors attribute for a car,
 - ▶ Cars with one color have a single value, whereas customized/sport cars might have two color values.
 - ▶ or a College_degrees attribute for a person.



Entity Types, Entity Sets, Attributes, and Keys

▣ Entities and Attributes

- Single-Valued versus Multivalued Attributes.
- **Multivalued:**
 - Similarly, one person may not have any college degrees,
 - another person may have one,
 - and a third person may have two or more degrees;
 - therefore, different people can have different numbers of values for the College degrees attribute.
- Should contain **lower and upper bounds** to limit the no. of values allowed for each individual entity.
- For example, the Colors attribute of a car may be restricted to have between one and two values, if we assume that a car can have two colors at most.

Entity Types, Entity Sets, Attributes, and Keys

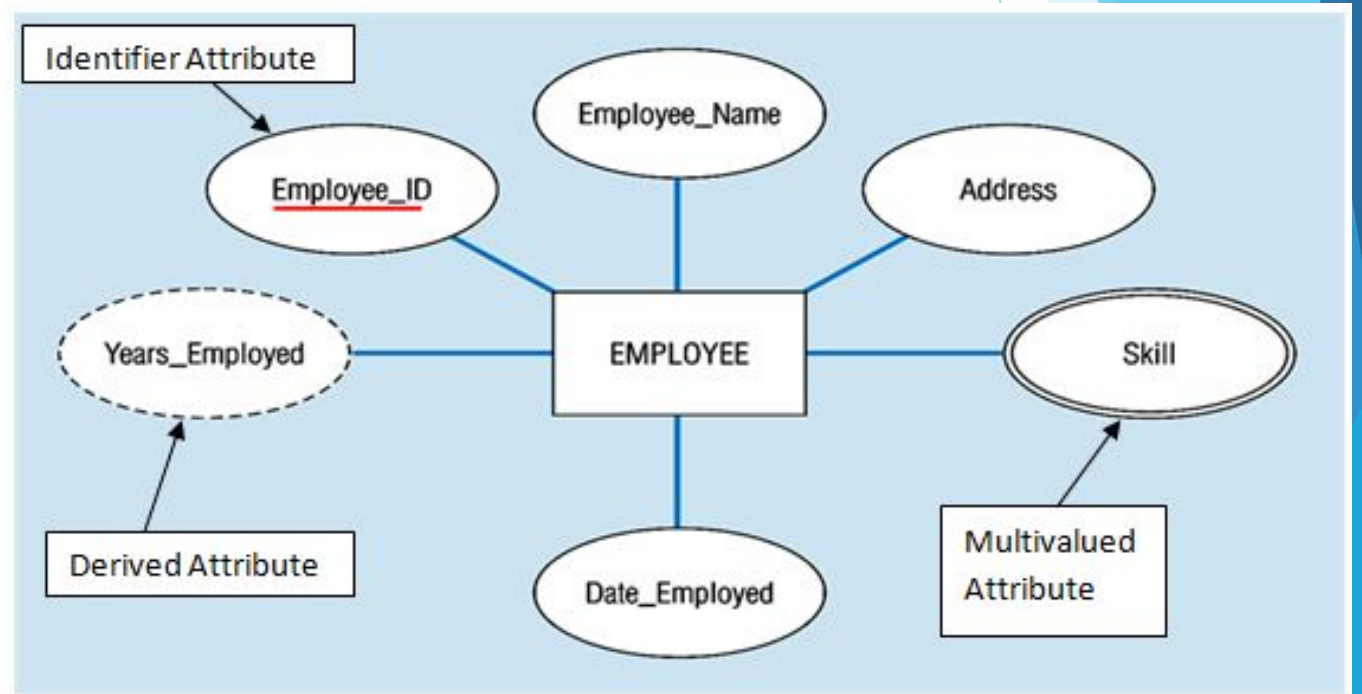
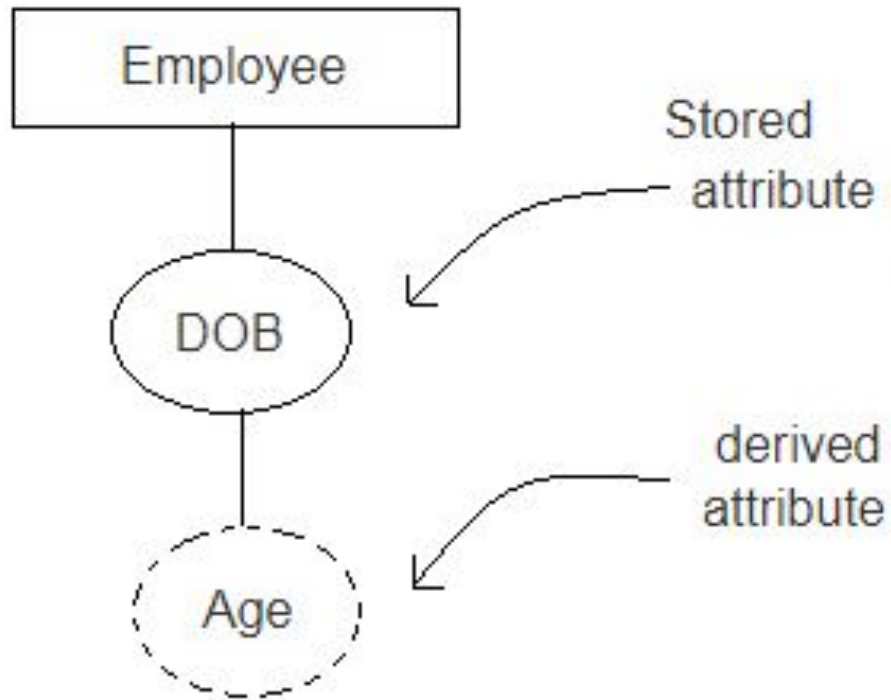
► Entities and Attributes

► Stored versus Derived Attributes

- In some cases, **two (or more) attribute values are related**
 - for example, **the Age and Birth_date attributes of a person.**
 - For a particular person entity, **Age = the current date - the person's Birth_date.**
- The Age attribute - derived attribute derivable by Birth_date
- The Birth_date - stored attribute.
- Some attribute values can be derived from related entities;
 - for example, an attribute **Number_of_employees of a DEPARTMENT entity** can be derived by counting the number of employees working for that department.

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ Stored versus Derived Attributes



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ NULL Values. (not APPLICABLE)
- ▶ In some cases, **a particular entity may not have an applicable value for an attribute.**
- ▶ For example: **Apartment_number attribute of an address** is valid for apartment buildings only and not to other types of residences, such as single-family homes.
- ▶ Similarly, a College_degrees attribute applies only to people with college degrees.
- ▶ **For such situations, a special value called NULL is created.**
- ▶ An address of a single-family home would have NULL for its Apartment_number attribute, and a person with no college degree would have NULL for College_degrees.
- ▶ Don't specify NOT NULL constraint over such attributes.

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes
- ▶ NULL Values. (UNKNOWN)
- ▶ **NULL can also be used if we do not know the value of an attribute for a particular entity**
 - ▶ for example, if we do not know the home phone number of 'John Smith' .
- ▶ The unknown category of NULL can be further classified into two cases.
 - ▶ the attribute value exists but is missing
 - ▶ for instance, if the Height attribute of a person is listed as NULL.
 - ▶ Not known whether the attribute value exists or not
 - ▶ for example, if the Home_phone attribute of a person is NULL.

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entities and Attributes

- ▶ Complex Attributes

- ▶ Composite and multivalued attributes can be nested.

- ▶ Complex Attributes: We can represent nesting by:

- ▶ grouping components of a composite attribute between parentheses ()
- ▶ and separating the components with commas,
- ▶ and by displaying multivalued attributes between braces { }.

- ▶ For example, **if a person can have more than one residence and each residence can have a single address and multiple phones**, an attribute Address_phone for a person can be specified as shown in Figure 3.5.4 Both Phone and Address are themselves composite attributes.

```
{Address_phone( {Phone(Area_code,Phone_number)},Address(Street_address  
(Number,Street,Apartment_number),City,State,Zip) )}
```

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entity Types, Entity Sets, Keys, and Value Sets

- ▶ Entity Types and Entity Sets

- ▶ **Entity:** each record is a real world object having real world existence

- ▶ **Entity type:**

- ▶ collection of entities having common attributes.
- ▶ Each entity type in the database is described by its name and attributes.

- ▶ **Entity set:**

- ▶ collection of one or more entities
- ▶ Entity type is the super set of all possible entity sets

- ▶ **Attributes :** properties that describes an entity

- ▶ **Domain:** set of permissive value for an attribute

Employee table

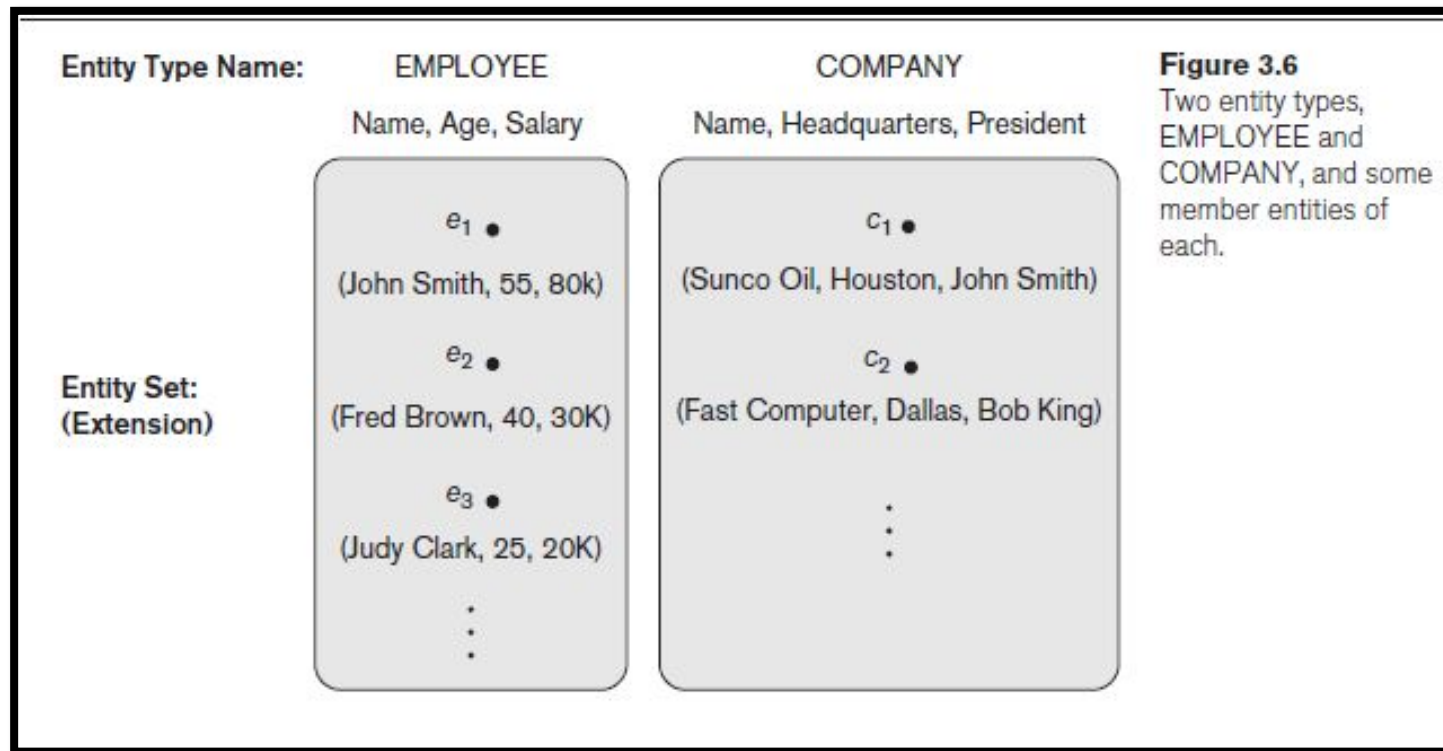
Entity type (describes
schema or intension)

ID	NAME	Email
1	Mohammad	m@gmail.com
2	Ahmed	a@gmail.com
3	Khan	k@gmail.com

Entity

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entity Types, Entity Sets, Keys, and Value Sets
- ▶ Entity Types and Entity Sets



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entity Types, Entity Sets, Keys, and Value Sets
- ▶ Representation in ER:
- ▶ An **entity type** in rectangular box enclosing the entity type name.
- ▶ **Attribute names** in ovals and are attached to their entity type by straight lines.
- ▶ **Composite attributes** are attached to their component attributes by straight lines.
- ▶ **Multivalued attributes** in double ovals.

Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entity Types, Entity Sets, Keys, and Value Sets

- ▶ Key Attributes of an Entity Type

- ▶ **key or uniqueness constraint on attributes:**

- ▶ applied on the entities of an entity type
- ▶ that uniquely identifies an entity

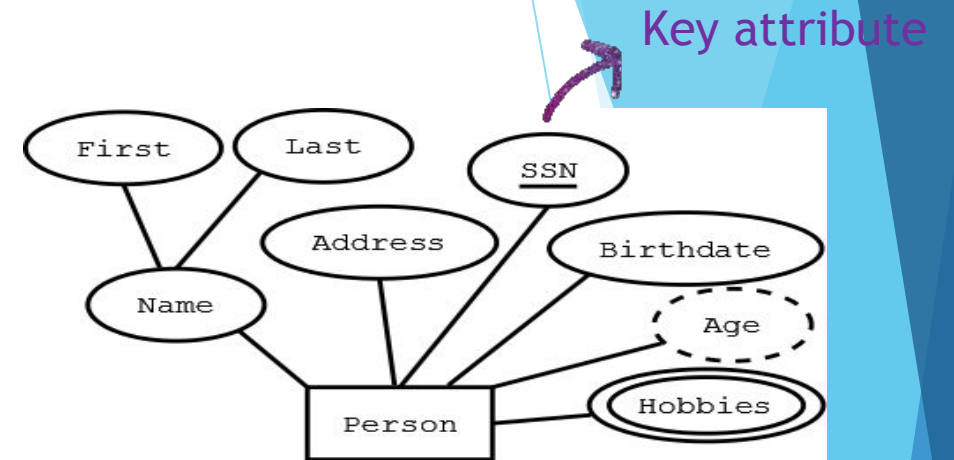
- ▶ it is a **constraint that prohibits any two entities from having the same value for the key attribute at the same time.**

- ▶ Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely.

- ▶ For example:

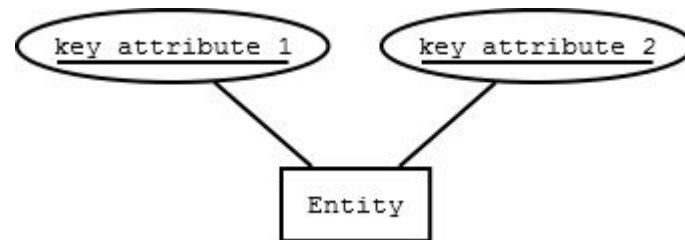
- ▶ the **Name** attribute is a key of the **COMPANY** entity type as no two companies are allowed to have the same name.
- ▶ The **SSN** (Social Security number) is a key attribute for **PERSON** entity type

- ▶ In ER diagrammatic notation, **each key attribute has its name underlined inside the oval.**



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Entity Types, Entity Sets, Keys, and Value Sets
- ▶ Key Attributes of an Entity Type
- ▶ Sometimes **several attributes together form a key**, meaning that the combination of the attribute values must be distinct for each entity.
- ▶ If a set of attributes possesses this property, the proper way to represent this in the ER model that we describe here is to define a composite attribute and designate it as a key attribute of the entity type.



Entity Types, Entity Sets, Attributes, and Keys

Entity Types, Entity Sets, Keys, and Value Sets

Key Attributes of an Entity Type

Some entity types have more than one key attribute.

For example,

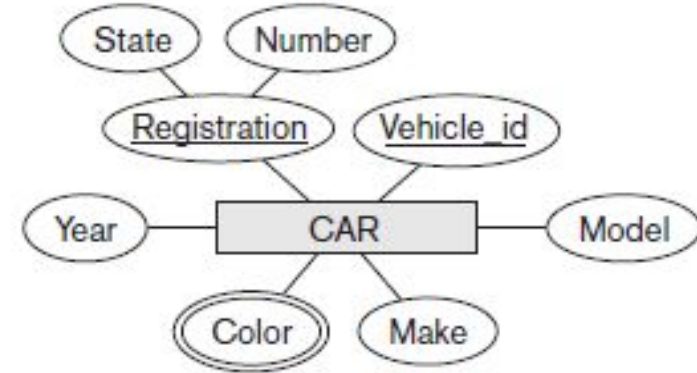
- each of the Vehicle_id and Registration attributes of the entity type CAR (Figure 3.7) is a key in its own right.

The Registration attribute is an example of a composite key formed from two simple component attributes, State and Number, neither of which is a key on its own.

An entity type may also have no key, in which case it is called a **weak entity type**

In our diagrammatic notation, **if two attributes are underlined separately, then each is a key on its own.**

(a)



(b)

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

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

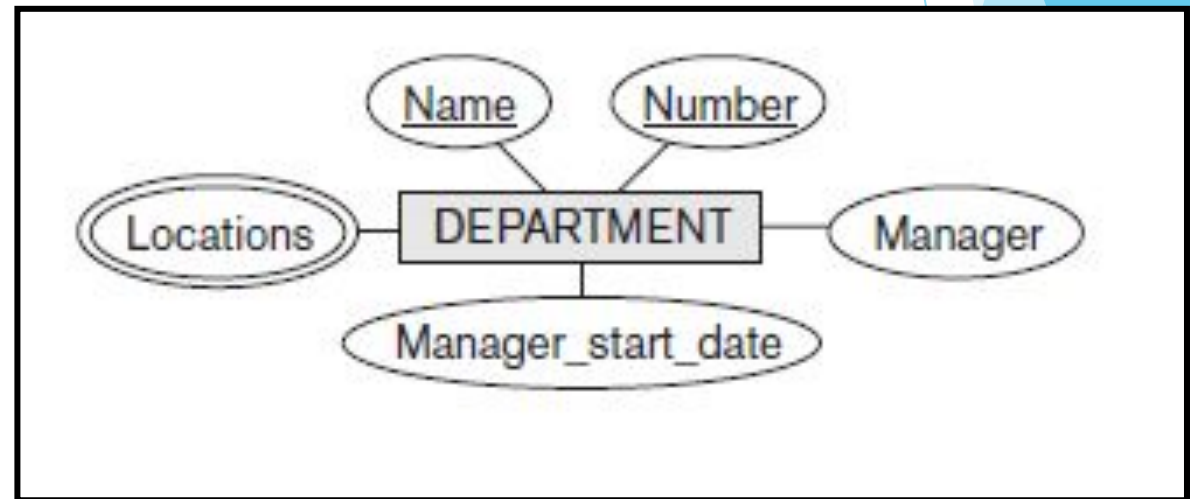
CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

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

⋮

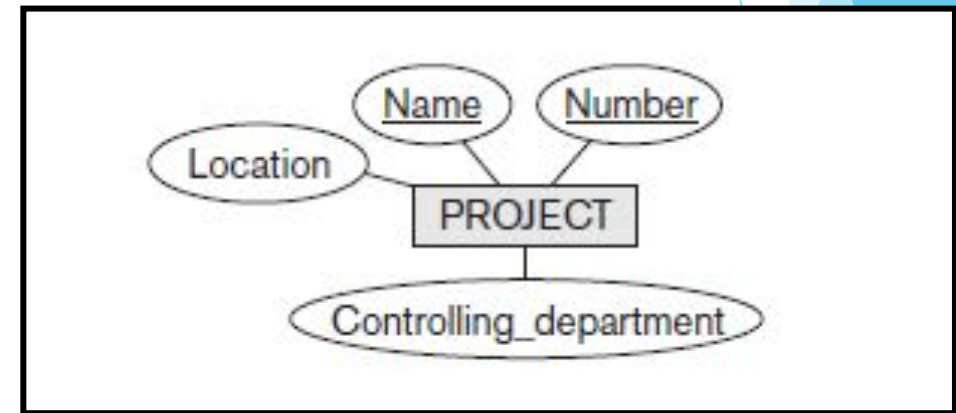
Entity Types, Entity Sets, Attributes, and Keys

- ▶ Initial Conceptual Design of the COMPANY Database
- ▶ We can now define the entity types for the COMPANY database, based on the requirements.
- ▶ 1. An entity type DEPARTMENT with attributes:
 - ▶ Name -> KEY ATTRIBUTE
 - ▶ Number -> KEY ATTRIBUTE
 - ▶ Locations, -> MULTI VALUED ATTRIBUTE
 - ▶ Manager,
 - ▶ Manager_start_date.



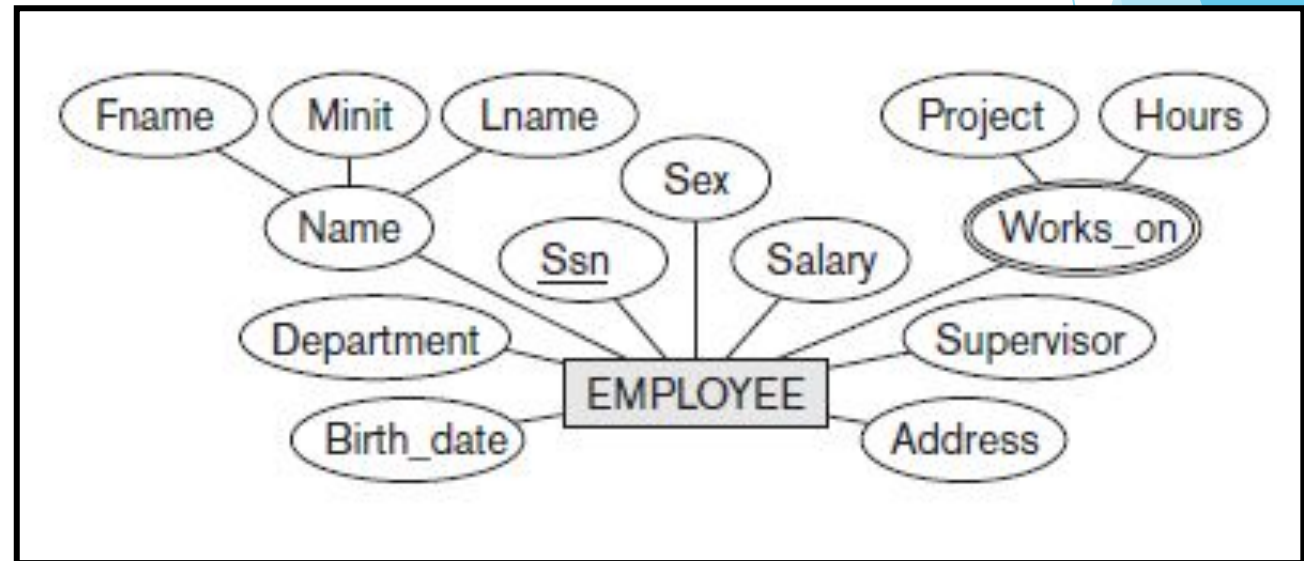
Entity Types, Entity Sets, Attributes, and Keys

- ▶ Initial Conceptual Design of the COMPANY Database
- ▶ We can now define the entity types for the COMPANY database, based on the requirements.
- ▶ 2. An entity type PROJECT with attributes:
 - ▶ Name -> KEY ATTRIBUTE
 - ▶ Number -> KEY ATTRIBUTE
 - ▶ Location
 - ▶ Controlling_department.



Entity Types, Entity Sets, Attributes, and Keys

- ▶ Initial Conceptual Design of the COMPANY Database
- ▶ We can now define the entity types for the COMPANY database, based on the requirements.
- ▶ 3. An entity type EMPLOYEE with attributes:
 - ▶ Name -> composite attribute
 - ▶ Ssn -> KEY ATTRIBUTE
 - ▶ gender,
 - ▶ Address,
 - ▶ Salary,
 - ▶ Birth_date,
 - ▶ Department,
 - ▶ Supervisor.
 - ▶ Works_on -> multivalued attribute

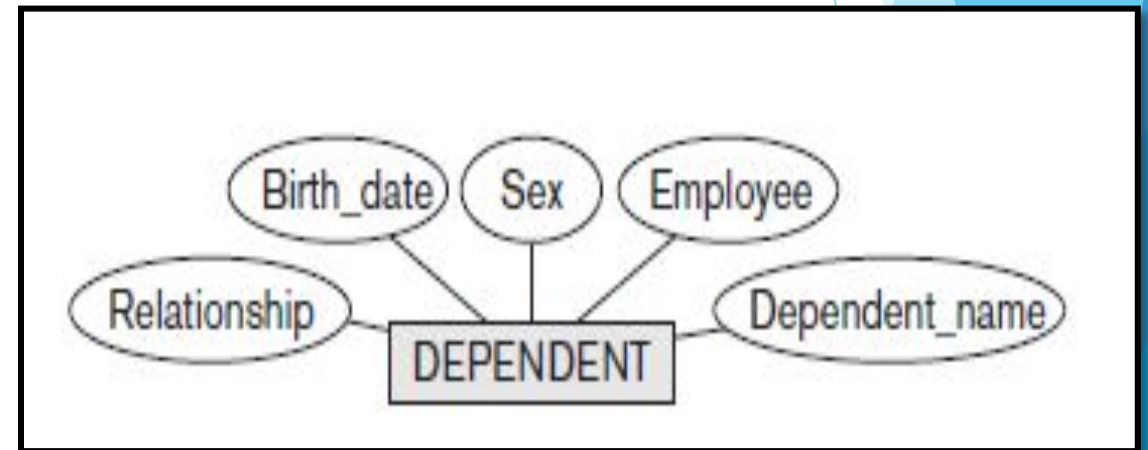


Entity Types, Entity Sets, Attributes, and Keys

- ▶ Initial Conceptual Design of the COMPANY Database
- ▶ Another requirement is that: **an employee can work on several projects, and the database has to store the number of hours per week an employee works on each project.**
- ▶ It can be represented by a **multivalued composite attribute of EMPLOYEE called Works_on** with the simple components (Project, Hours).
- ▶ Alternatively, it can be represented as **a multivalued composite attribute of PROJECT called Workers** with the simple components (Employee, Hours).

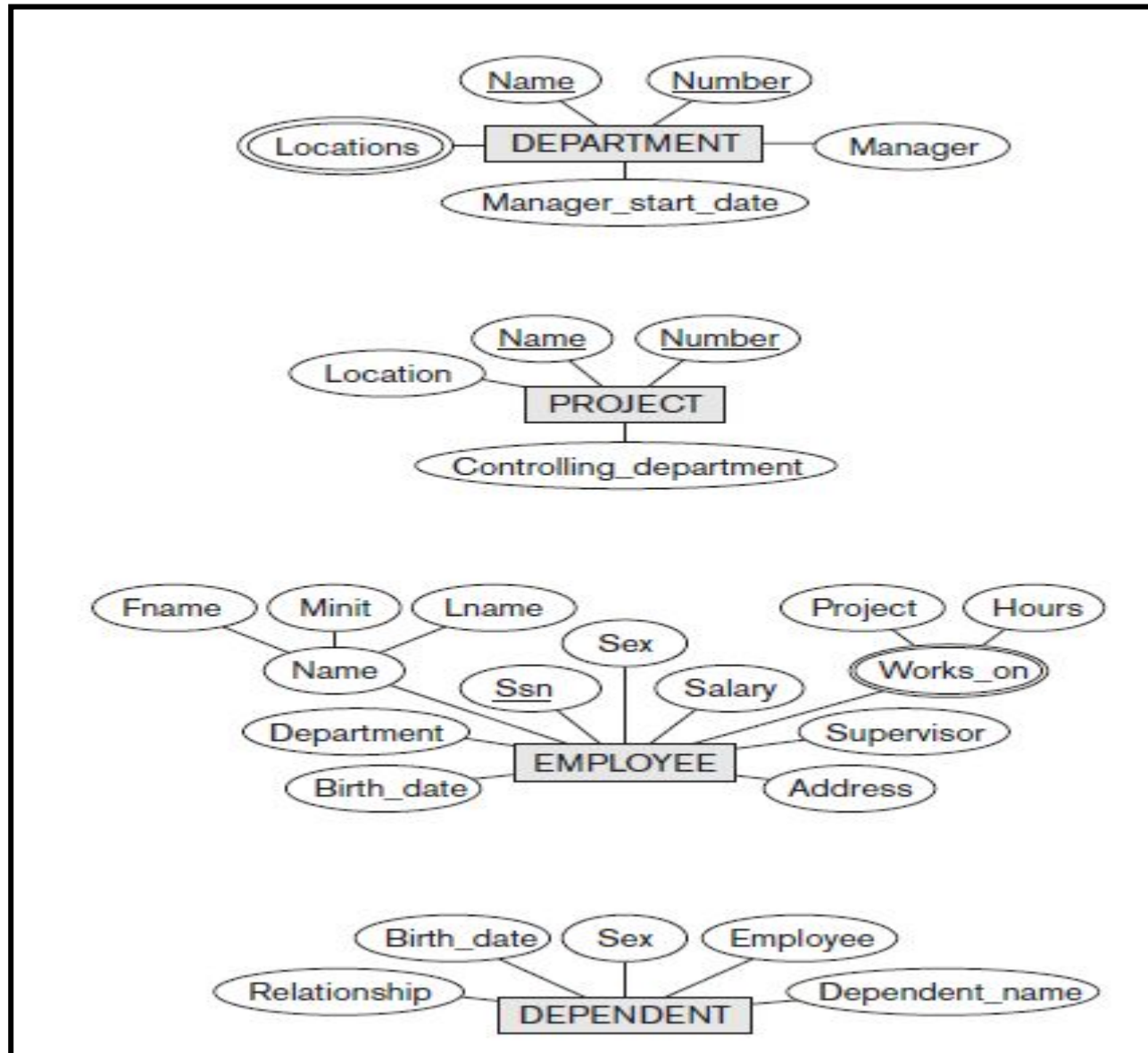
Entity Types, Entity Sets, Attributes, and Keys

- ▶ Initial Conceptual Design of the COMPANY Database
- ▶ We can now define the entity types for the COMPANY database, based on the requirements.
- ▶ 4. An entity type DEPENDENT with attributes
 - ▶ EmployeeNo,
 - ▶ Dependent_name,
 - ▶ Gender,
 - ▶ Birth_date,
 - ▶ Relationship (to the employee).



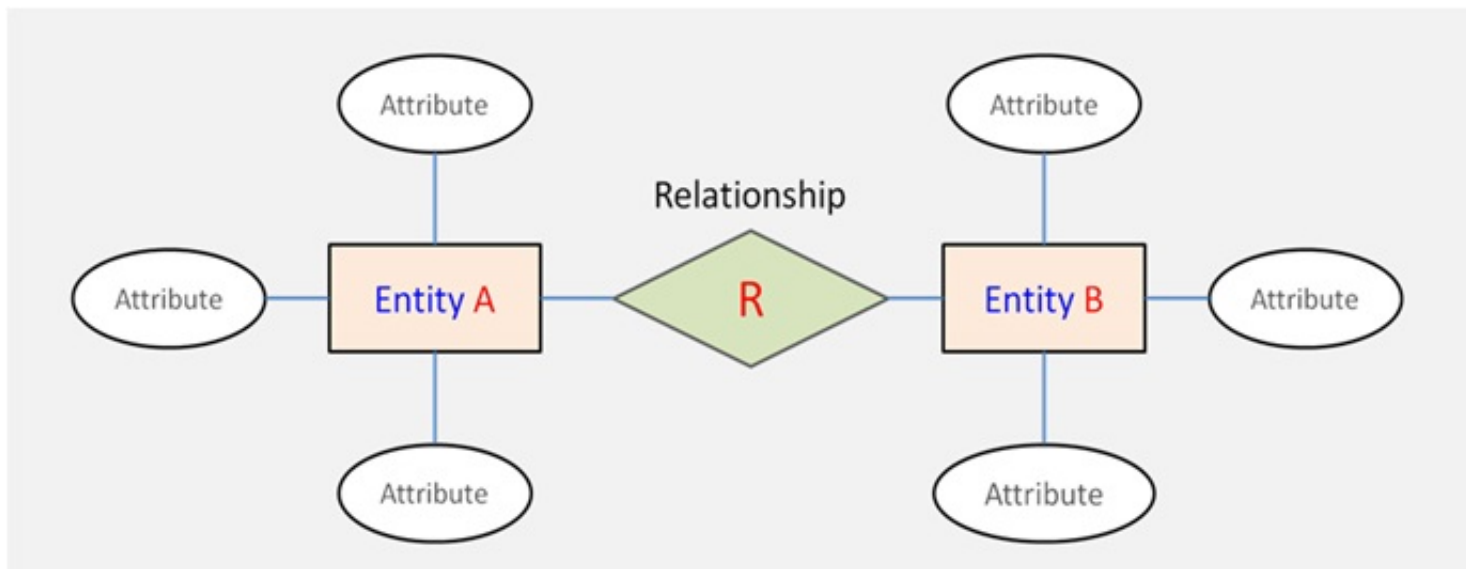
Entity Types, Entity Sets, Attributes, and Keys

► Initial Conceptual Design of the COMPANY Database



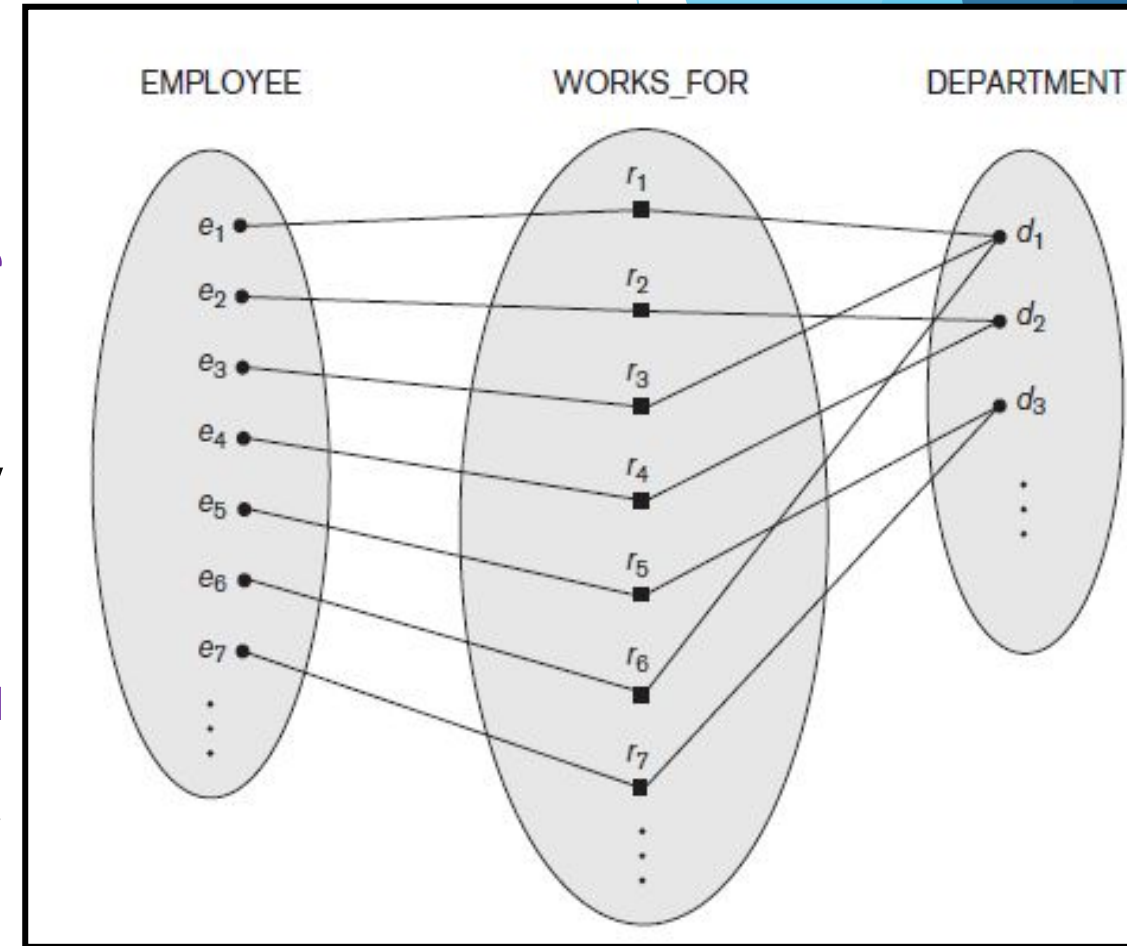
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Types, Sets, and Instances
- ▶ A **relationship type R** among n entity types E_1, E_2, \dots, E_n defines a set of associations among entities from these entity types.
- ▶ Each of the entity types E_1, E_2, \dots, E_n is said to participate in the relationship type R .



Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Types, Sets, and Instances
- ▶ For example:
- ▶ Consider a relationship type **WORKS_FOR**
- ▶ between the two entity types **EMPLOYEE** and **DEPARTMENT**,
- ▶ which associates each employee with the department for which the employee works.
- ▶ Each relationship instance in the relationship set **WORKS_FOR** associates one **EMPLOYEE** entity and one **DEPARTMENT** entity.
- ▶ In ER diagrams,
 - ▶ Relationship types are displayed as diamond-shaped boxes,
 - ▶ which are connected by straight lines to the rectangular boxes representing the participating entity types.
- ▶ The **relationship name** is displayed in the diamond-shaped box.



Relationship Types, Relationship Sets, Roles & Structural Constraints

Relationship Degree, Role Names, and Recursive Relationships

Degree of a Relationship Type

degree of a relationship type = No. of participating entities.

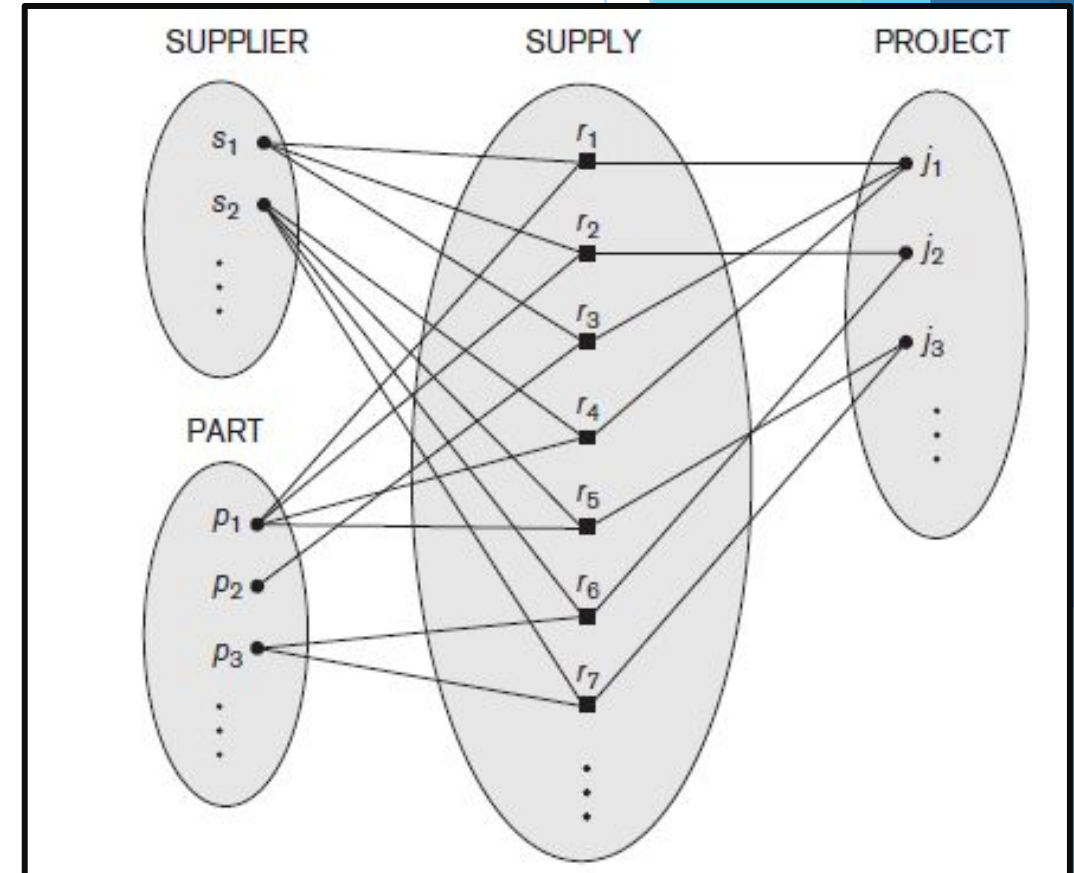
Degree of WORKS_FOR relationship = 2.

Binary: A relationship type of degree two

Ternary: A relationship type of degree three

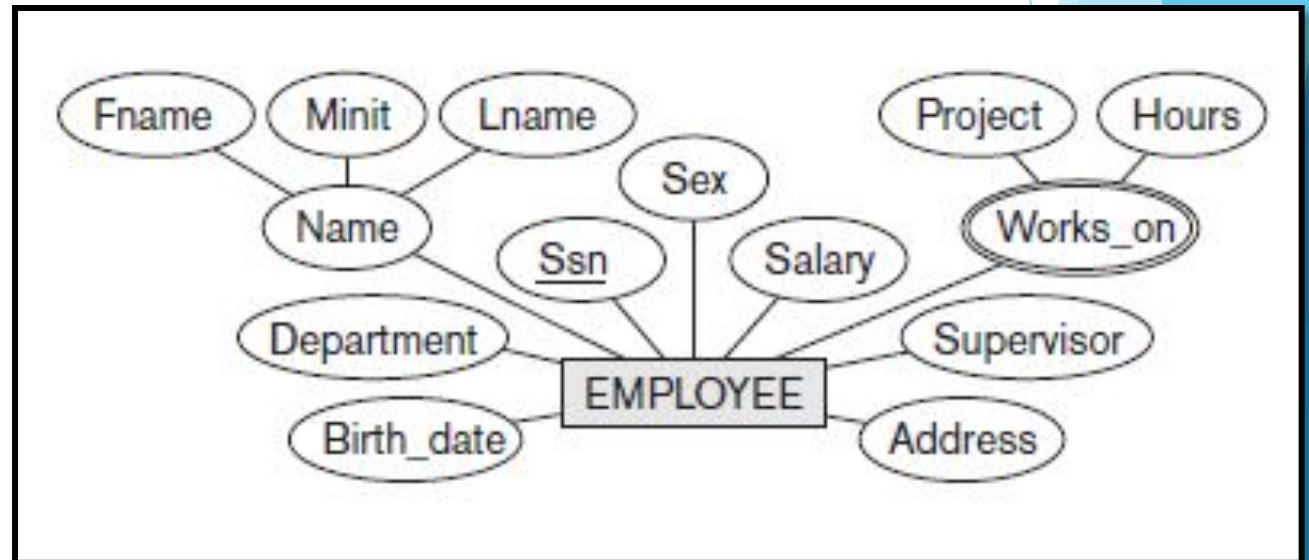
An example of a ternary relationship is SUPPLY, where each relationship instance associates three **entities**

- ▶ a supplier s ,
- ▶ a part p ,
- ▶ and a project j
- ▶ whenever s supplies part p to project j .



Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Degree, Role Names, and Recursive Relationships
- ▶ **Relationships as attributes**
- ▶ It is sometimes easy to think of binary relations as attributes in an entity set
- ▶ As we have done in the following diagram.



Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Degree, Role Names, and Recursive Relationships
- ▶ *Role Names and Recursive Relationships*
- ▶ Each entity type that participates in a relationship type plays a particular role in the relationship.
- ▶ **role name:** mentions the role that a participating entity from the entity type plays in each relationship instance, and it helps to explain what the relationship means.
- ▶ For example, in the WORKS_FOR relationship type, EMPLOYEE **plays the role of employee/worker** and DEPARTMENT **plays the role of department/employer**.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Degree, Role Names, and Recursive Relationships
- ▶ **Role Names and Recursive Relationships**
- ▶ In some cases, the **same entity type participates more than once in a relationship type in different roles**.
- ▶ In such cases the **role name becomes essential** for distinguishing the meaning of the role that each participating entity plays.
- ▶ Such relationship types are called **recursive relationships** or **self-referencing relationships**.
- ▶ The SUPERVISION relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same EMPLOYEE entity set. Hence, the EMPLOYEE entity type participates twice in SUPERVISION: once in the role of supervisor (or boss), and once in the role of supervisee (or subordinate).

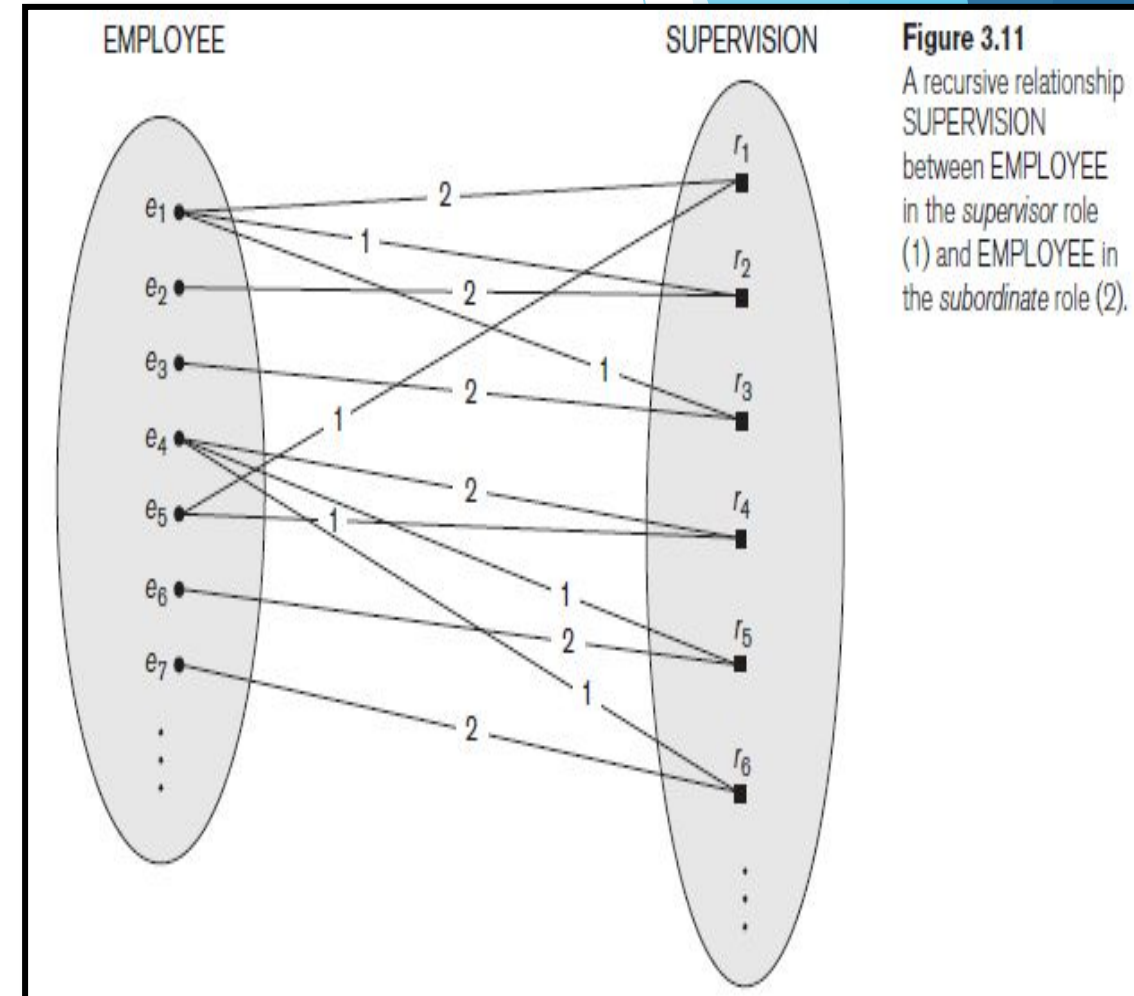
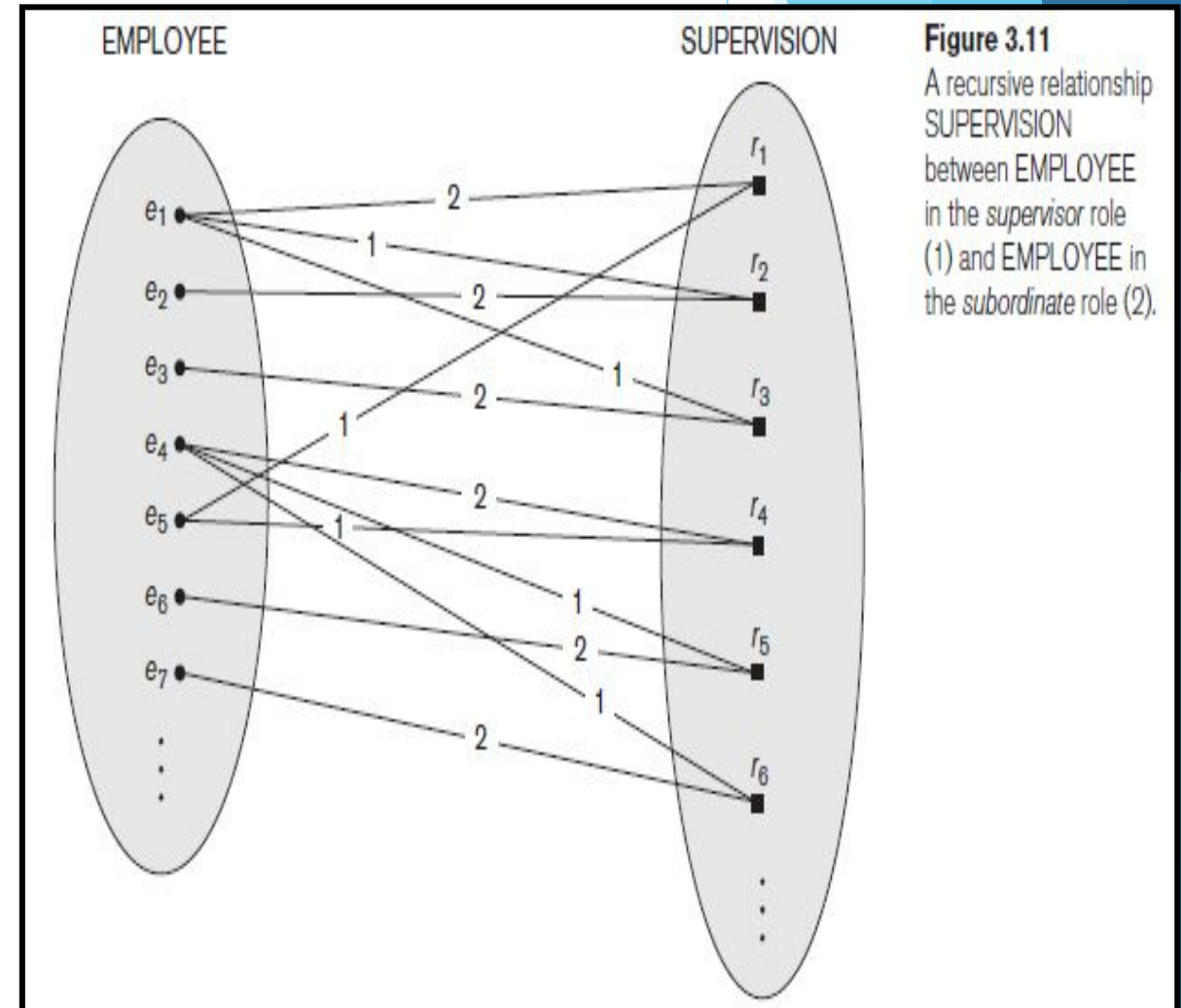


Figure 3.11

A recursive relationship SUPERVISION between EMPLOYEE in the supervisor role (1) and EMPLOYEE in the subordinate role (2).

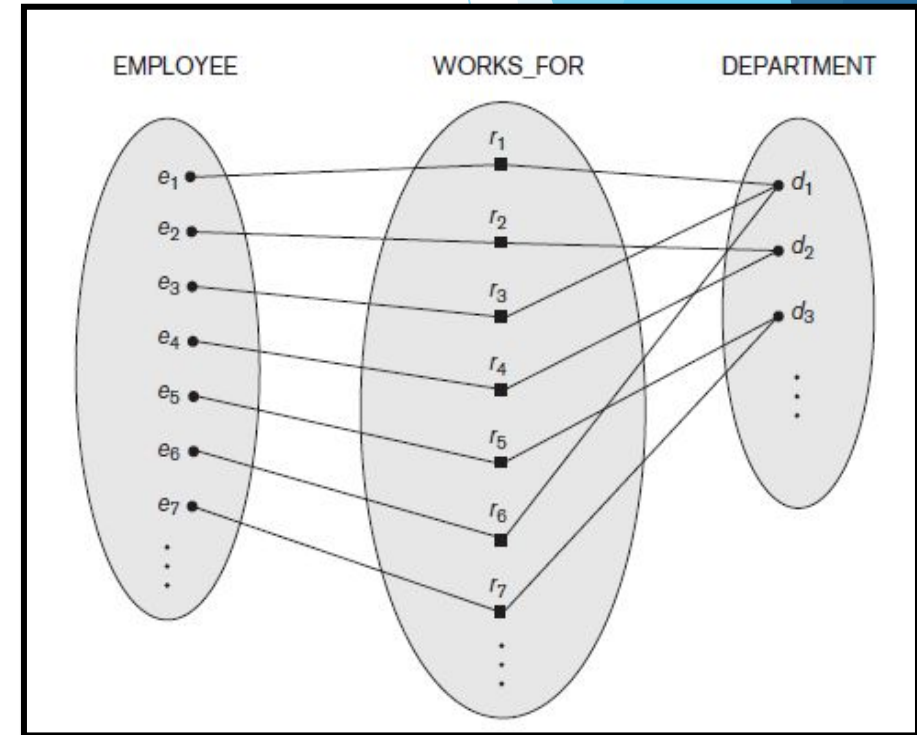
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Relationship Degree, Role Names, and Recursive Relationships
- ▶ **Role Names and Recursive Relationships**
- ▶ In Figure 3.11
 - ▶ the lines marked '1' represent the supervisor role
 - ▶ The lines marked '2' represent the supervisee role
 - ▶ hence,
 - ▶ e1 supervises e2 and e3, and works under e5
 - ▶ e4 supervises e6 and e7, and works under e5
 - ▶ e5 supervises e1 and e4.
- ▶ In this example, each relationship instance must be connected with two lines, one marked with '1' (supervisor) and the other with '2' (supervisee).



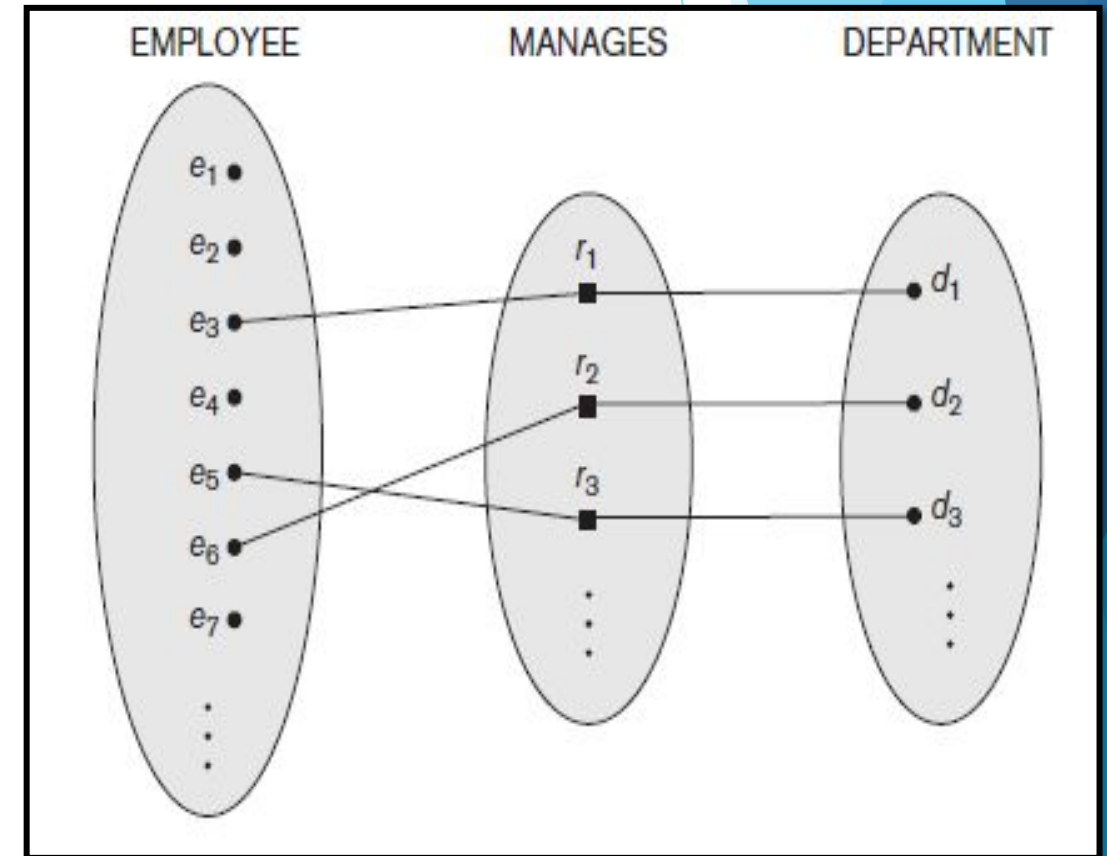
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- Binary Relationship constraints
- Type 1: **Cardinality Ratios for Binary Relationships**
- **cardinality ratio for a binary relationship:**
 - the maximum number of relationship instances that an entity can participate in.
- For example,
 - Requirement: each employee must work for 1 department
 - in the **WORKS_FOR** binary relationship type, **DEPARTMENT:EMPLOYEE** is of cardinality ratio **1:N**,
 - meaning that each department can be related to (that is, employs) any number of employees (N), but an employee can be related to (work for) at most one department.
- This means that for this particular relationship type WORKS_FOR, a particular department entity can be related to any number of employees
- (N indicates there is no maximum number).



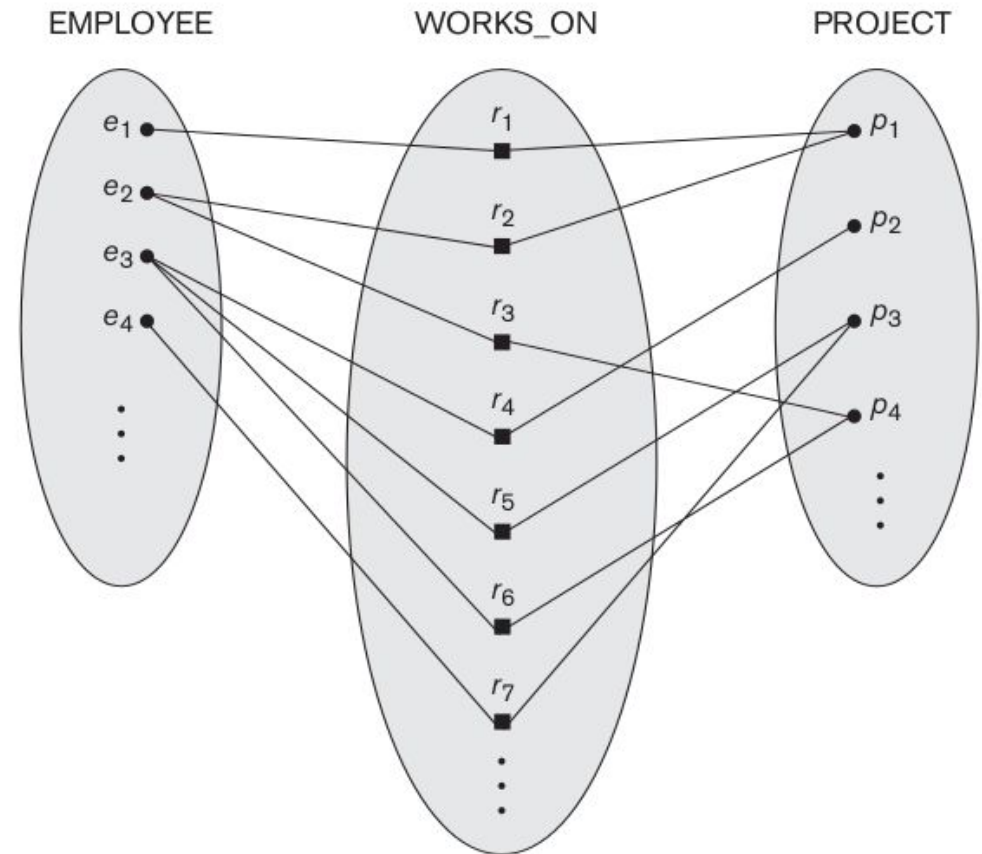
Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Cardinality Ratios for Binary Relationships**
- ▶ The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and N:N.
- ▶ Requirement: an employee can manage at most one department and a department can have at most one manager.
 - ▶ An example of a 1:1 binary relationship
 - ▶ **MANAGES** which relates a department entity to the employee who manages that department.



Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Cardinality Ratios for Binary Relationships**
- ▶ Requirement: an employee can work on several projects and a project can have several employees.
 - ▶ The relationship type WORKS_ON is of cardinality ratio N:N
 - ▶ e2 is working on p1 and p4
 - ▶ p4 has 2 employees e2 and e3



Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ Type 2: **Participation Constraints and Existence Dependencies**
- ▶ The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- ▶ This constraint specifies the minimum number of relationship instances that each entity can participate in
- ▶ Sometimes called as the minimum cardinality constraint.
- ▶ There are two types of participation constraints
 - ▶ **Total**
 - ▶ **Partial.**

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Participation Constraints and Existence Dependencies**
- ▶ If a company policy states that
 - ▶ **Requirement: every employee must work for a department**
 - ▶ **Then An employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.**
- ▶ Thus, the **participation of EMPLOYEE in WORKS_FOR is called total participation**, meaning that every entity in the total set of employee entities must be related to a department entity via WORKS_FOR.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Participation Constraints and Existence Dependencies**
- ▶ **Total participation is also called existence dependency.**
- ▶ **Requirement: It is not expected that every employee manages a department,**
 - ▶ so the **participation of EMPLOYEE in the MANAGES relationship type is partial,**
 - ▶ meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.
- ▶ **structural constraints of a relationship type = cardinality ratio + participation constraints,**
- ▶ In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Attributes of Relationship Types**
- ▶ Relationship types can also have attributes, similar to those of entity types.
- ▶ For example,
 - ▶ record the number of hours per week that a particular employee works on a particular project, in WORKS_ON relationship type
- ▶ Another example
 - ▶ Record the Start_date on which a manager started managing a department for the MANAGES relationship type

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Binary Relationship constraints
- ▶ **Attributes of Relationship Types**
- ▶ Notice that attributes of **1:1 or 1:N** relationship types can be migrated to one of the participating entity types.
 - ▶ For example, the Start_date attribute for the MANAGES relationship can be an attribute of either EMPLOYEE (manager) or DEPARTMENT, although conceptually it belongs to MANAGES. This is because MANAGES is a 1:1 relationship, so every department or employee entity participates in at most one relationship instance.
- ▶ Hence, the value of the Start_date attribute can be determined separately, either by the **participating department entity or by the participating employee (manager) entity.**

Relationship Types, Relationship Sets, Roles, and Structural Constraints

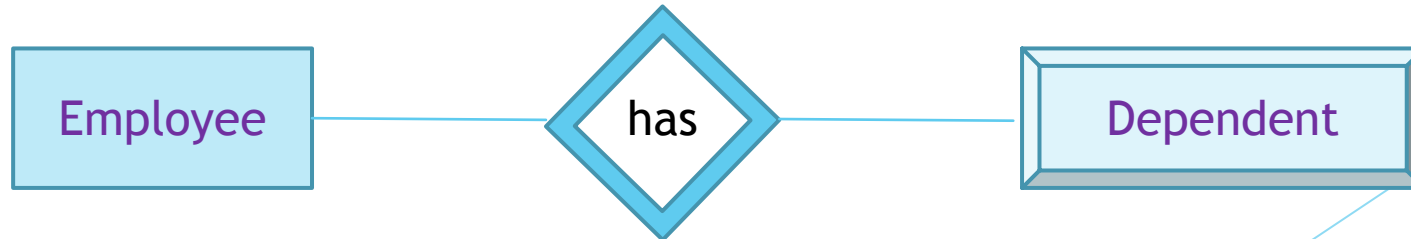
- ▶ Binary Relationship constraints
- ▶ *Attributes of Relationship Types*
- ▶ For a 1:N relationship type, a relationship attribute can be migrated only to the entity type on the **N-side** of the relationship.
- ▶ For example, in Figure 3.9, if the WORKS_FOR relationship also has an attribute Start_date that indicates when an employee started working for a department, this attribute can be included as an attribute of **EMPLOYEE**.
- ▶ This is because each employee works for at most one department, and hence participates in at most one relationship instance in WORKS_FOR, but a department can have many employees, each with a different start date.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- ▶ Constraints on Binary Relationship Types
- ▶ **Attributes of Relationship Types**
- ▶ For M:N (many-to-many) relationship types, some attributes may be **determined by the combination of participating entities** in a relationship instance, not by any single entity.
- ▶ Such attributes must be specified as relationship attributes.
- ▶ An example is the Hours attribute of the M:N relationship WORKS_ON (Figure 3.13); the number of hours per week an employee currently works on a project is determined by an employee-project combination and not separately by either entity.

Weak Entity Types

- ▶ Entity types that do not have key attributes of their own are called **weak entity types**.
- ▶ Entities belonging to a weak entity type are identified by being related to **specific entities from another entity type in combination with one of their attribute values**.
- ▶ We call this other entity type the identifying or **owner entity type**, and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type.



Owner entity type

Identifying relationship

Weak Entity Types

- ▶ A weak entity type always has a total participation constraint (existence dependency) W.r.t its identifying relationship because a **weak entity cannot be identified without an owner entity**.
- ▶ However, not every **total participation** results in a weak entity type.
 - ▶ For example,
 - ▶ a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity.

Weak Entity Types

- ▶ Consider the entity type **DEPENDENT**, related to **EMPLOYEE**, which is used to keep track of the dependents of each employee via a 1:N relationship.
- ▶ In our example, the attributes of **DEPENDENT** are **Name** (the first name of the dependent), **Birth_date**, **gender** and **Relationship** (to the employee).
- ▶ Two dependents of two distinct employees may, by chance, have the same values for Name, Birth_date, gender, and Relationship, but they are still distinct entities.
- ▶ They are identified as distinct entities only after determining the **particular employee entity to which each dependent is related**.
- ▶ Each employee entity is said to own the dependent entities that are related to it.

Weak Entity Types

- ▶ A weak entity type normally has a **partial key**, which is the attribute that can uniquely **identify weak entities** that are related to the same owner entity.
- ▶ In our example,
 - ▶ if we assume that no **two dependents of the same employee** ever have the same first name, the attribute Name of DEPENDENT is the **partial key**.
- ▶ In ER diagrams representation, weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines respectively.
- ▶ The partial key attribute is **underlined with a dashed or dotted line**.

Weak Entity Types

- ▶ Weak entity types can sometimes be represented as complex (composite, multivalued) attributes.
- ▶ In the preceding example, we could specify a multivalued attribute Dependents for EMPLOYEE, which is a **multivalued composite attribute** with the component attributes **Name, Birth_date, gender, and Relationship**.

Refining the ER Design for the COMPANY Database

In our example, we specify the following relationship types:

- MANAGES,
 - which is a 1:1(one-to-one) relationship type between EMPLOYEE and DEPARTMENT.
 - EMPLOYEE participation is partial.
 - The attribute Start_date is assigned to this relationship type.

- WORKS_FOR,
 - a 1:N (one-to-many) relationship type between DEPARTMENT and EMPLOYEE.
 - Both participations are total.

- CONTROLS,
 - a 1:N relationship type between DEPARTMENT and PROJECT.
 - The participation of PROJECT is total,
 - whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.

Refining the ER Design for the COMPANY Database

In our example, we specify the following relationship types:

■ SUPERVISION,

- a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).
- Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.

■ WORKS_ON,









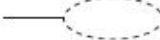


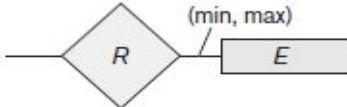
- an M:N (many-to-many) relationship type with attribute Hours, after the users indicate that a project can have several employees working on it.
- Both participations are determined to be total.

■ DEPENDENTS_OF,

- a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT.
- The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

ER Diagrams, Naming Conventions, and Design Issues

► Summary of 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

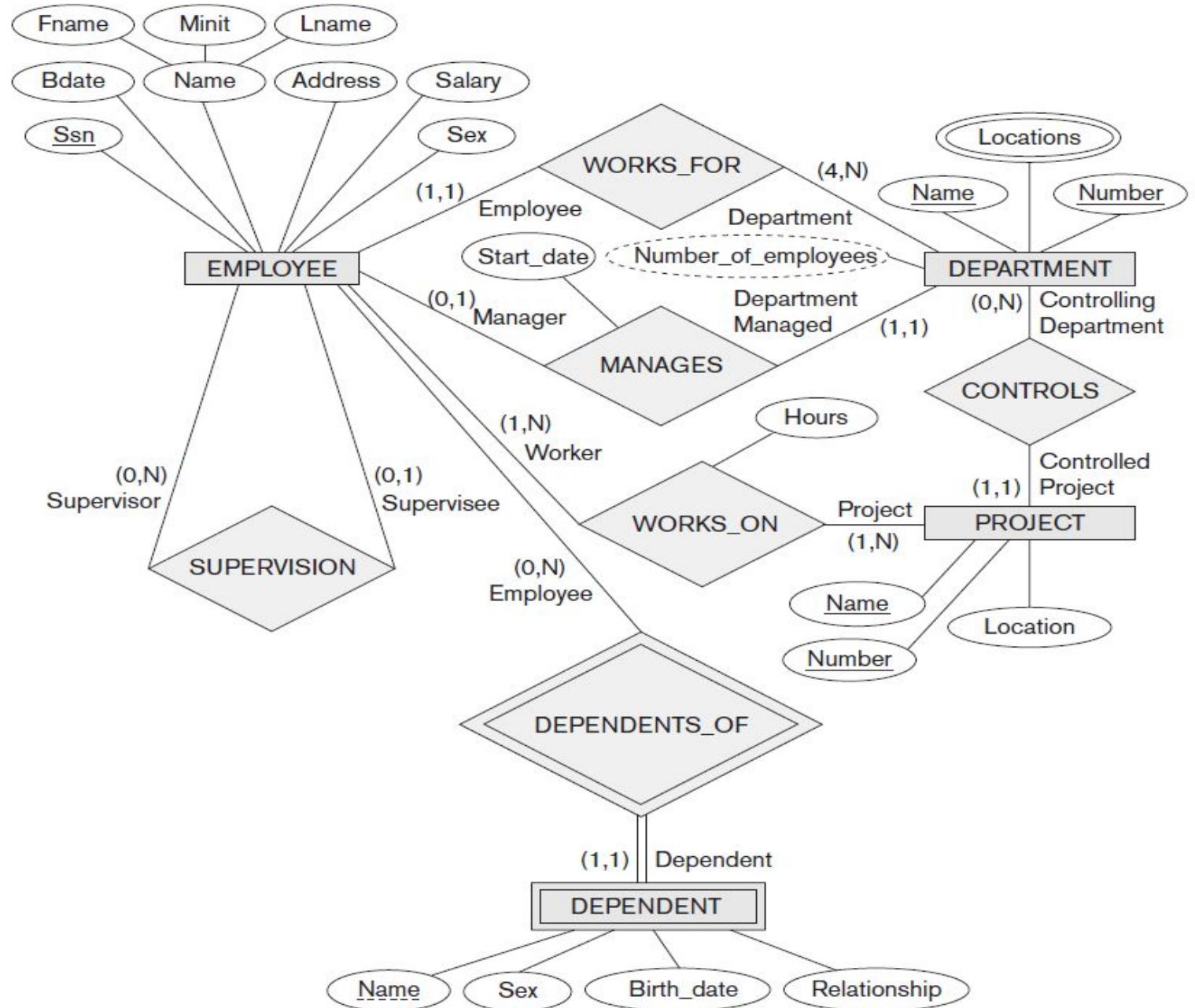
ER Diagrams, Naming Conventions, and Design Issues

- ▶ *Proper Naming of Schema Constructs*
- ▶ Choose names that convey, as much as possible. (meaningful names)
- ▶ Use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type.
 - ▶ Entity type and relationship type names are in uppercase letters,
 - ▶ Attribute names have their initial letter capitalized,
 - ▶ and role names are in lowercase letters.

A Sample Database Application

- ▶ The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- ▶ A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- ▶ The database will store each employee's name, Social Security number, address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- ▶ The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

ER Diagrams, Naming Conventions, and Design Issues



A Sample Database Application

