

UNIT – III chapter 1
For Advanced SQL see PPTs

UNIT – III chapter 2

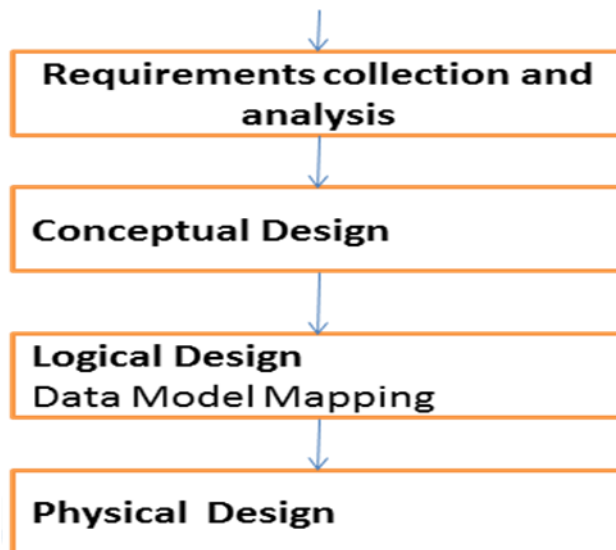
Data Modelling Using the Entity..Relationship Model

Database application refers to a particular database and the associated programs that implement the database queries and updates.

For example, a BANK database application that keeps track of customer accounts would include programs that implement database updates corresponding to customers making deposits and withdraw, also These programs provide user-friendly graphical user interfaces (GUIs) utilizing forms and menus for the end users of the application-the bank tellers, in this example. Hence, part of the database application will require the design, implementation, and testing of these **application programs**.

Entity-Relationship (ER) model, which is a popular high, level conceptual data model. This model and its variations are used for the conceptual design of database applications

Phases of Database Design



Above figure shows a simplified description of the database design process.

Requirements collection and analysis:

- In this step, the database designers interview prospective database users to understand and document their data requirements.
- The result of this step is a concisely written set of users' requirements.
- These requirements should be specified in as detailed and complete a form as possible

Conceptual Design

- Once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database; using a high-level conceptual data model this step is called conceptual design.
- The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed by using the ER Model
- This approach enables the database designers to concentrate on specifying the properties of the data, without being concerned with storage details.

Logical Design

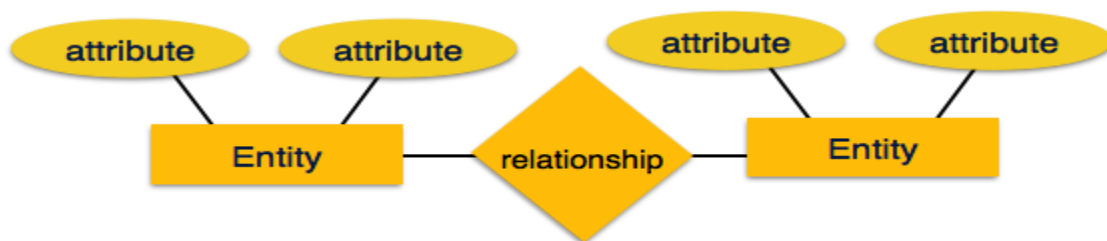
- The next step in database design is the actual implementation of the database, using a commercial DBMS.
- Most current commercial DBMSs use an implementation data model such as the relational or the object-relational database model.
- In this phase the conceptual schema is transformed from the high-level data model into the implementation data model.
- This step is called logical design or data model mapping, and its result is a database schema in the implementation data model of the DBMS.

Physical Design

- The last step is the physical design phase, during which the internal storage structures, indexes, access paths, and file organizations for the database files are specified.
- In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

ER (Entity-Relationship) Model

- **Entity-Relationship model** is a popular high, level conceptual data model. This model and its variations are frequently used for the conceptual design of Database applications
- The ER model describes data as *entities*, *relationships*, and *attributes*.
- ER Model is best used for the conceptual design of database.
- ER Model is based on: **Entities** and their *attributes*, **Relationships** among entities



Entity:

- An entity, which is a "thing" 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 example, a company, a job, or a university course)

- In the University database context, an individual *student*, *faculty member*, a *class room*, a *course* are entities
- An entity represents some "thing" (in the mini world) that about which we want to maintain some data. An entity could represent a physical object (e.g., house, person, automobile, and widget) or a less tangible concept (e.g., company, job, academic course).

Attributes

- Attribute describes property or characteristics of an entity
- These are the properties of the entity.
- For example, an employee entity may be described by the employee's name, age, address, salary, and job. A particular entity will have a value for each of its attribute

Types of Attributes

1. Simple versus composite
2. single-valued versus multivalued
3. stored versus derived.

1. Composite versus Simple (Atomic) Attributes

Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.

For example, the Address attribute of the employee entity can be subdivided into StreetAddress, City, State, and Zip,³ with the values "2311 Kirby," "Houston," "Texas," and "77001."

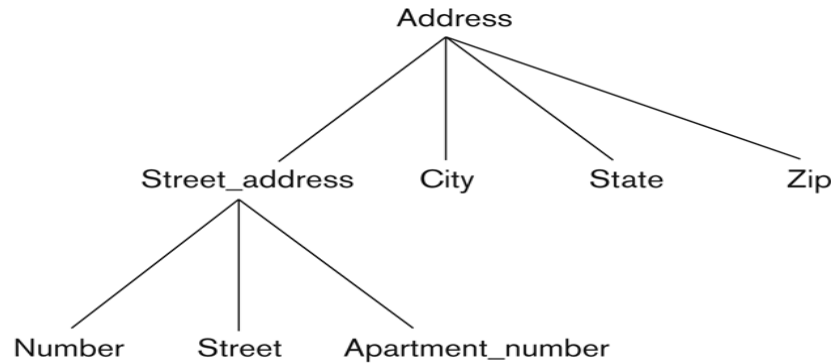


Figure 3.4
A hierarchy of composite attributes.

Simple Attributes: Attributes that are not divisible are called **simple** or **atomic** attributes.

Ex: Zipcode

2. Single-Valued versus Multivalued Attributes

Single-valued attribute is the attribute which is having only single value.

Most attributes have a single value for a particular entity; such attributes are called **single-valued**.

For example, Age, is a single-valued attribute of a person.

Multi-valued attribute is the attribute which is having more than one value.

An attribute can have a set of values for the same entity-for example, Colors attribute for a car, or a phno attribute for a person.

3. Stored versus Derived Attributes.

Derived attribute is one whose value can be calculated/derived from the values of other attributes. The Age and BirthDate attributes of a person. For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's BirthDate. The Age attribute is hence called a **derived attribute** and is said to be **derivable from** the BirthDate attribute, which is called a **stored attribute**.

Entity Types and Entity Sets:

- An **entity type** defines a *collection* (or *set*) of entities that have the same attributes.
- Each entity type in the database is described by its name and attributes.
- A database usually contains groups of entities that are similar.
- For example, a company employing hundreds of employees may want to store similar information concerning each of the employees.
- These employee entities share the same attributes, but each entity has *its own value(s)* for each attribute.
- The collection of all entities of a particular entity type in the database at any point in time is called an **entity set**.
- The entity set is usually referred to using the same name as the entity type.
- An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.
- An entity type describes the schema or intension for a *set of entities* that share the same structure.

Key Attributes of an Entity Type

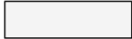

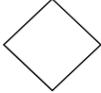




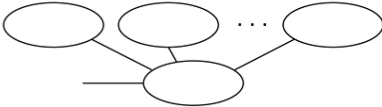



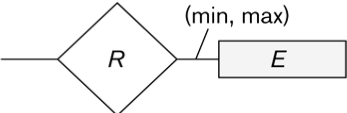
- An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.
- An entity type usually has an attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely.
- For example, the Name attribute is a key of the COMPANY entity type because no two companies are allowed to have the same name.
- For the Employee entity type typical key attribute is Empid.
- Specifying that an attribute is a key of an entity type means that the preceding uniqueness property must hold for *every entity set* of the entity type. Hence, it is a constraint that prohibits any two entities from having the same value for the key attribute at the same time.

Domains of Attributes:

- Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute.
- For each individual entity we can specify the value set.
- For the Name attribute as being the set of strings of alphabetic characters separated by blank characters, and so on.
- Value sets are not displayed in ER diagrams. Value sets are typically specified using the basic data types available in most programming languages, such as integer, string, boolean, float, enumerated type, subrange, and so on.

Notations of 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

Example Database Application: COMPANY

Initial Conceptual Design of COMPANY database

Suppose that Requirements Collection and Analysis results in the following (informal) description of the COMPANY mini world:

The company is organized as a collection of **departments**.

- Each department
 - has a unique name
 - has a unique number
 - is associated with a set of locations
 - has a particular employee who acts as its manager (and who assumed that position on some date)
 - has a set of employees assigned to it
 - controls a set of projects
- Each project
 - has a unique name
 - has a unique number
 - has a single location
 - has a set of employees who work on it
 - is controlled by a single department
- Each employee
 - has a name
 - has a SSN that uniquely identifies her/him
 - has an address
 - has a salary
 - has a sex
 - has a birthdate
 - has a direct supervisor
 - has a set of dependents
 - is assigned to one department
 - works some number of hours per week on each of a set of projects (which need not all be controlled by the same department)
- Each dependent
 - has first name
 - has a sex
 - has a birthdate
 - is related to a particular employee in a particular way (e.g., child, spouse, pet)
 - is uniquely identified by the combination of her/his first name and the employee of which (s)he is a dependent

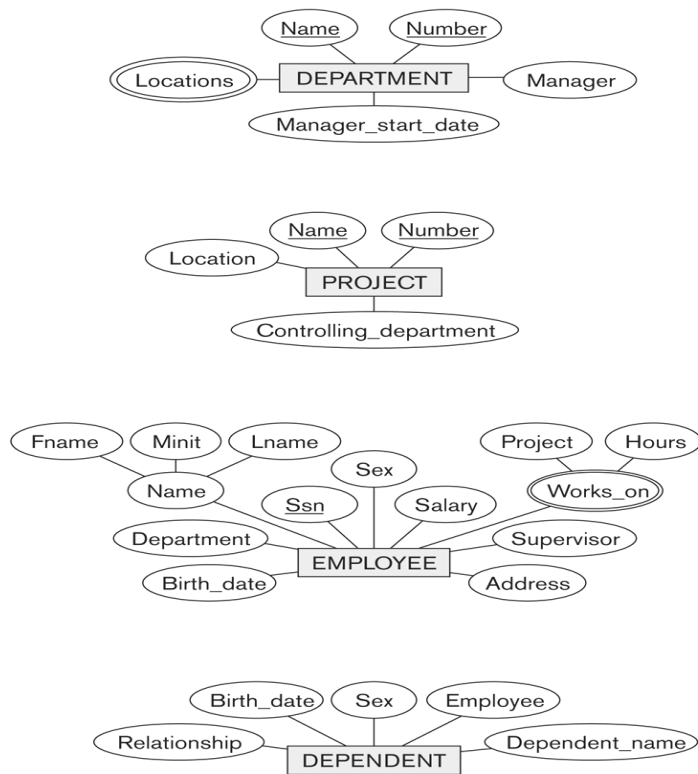


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

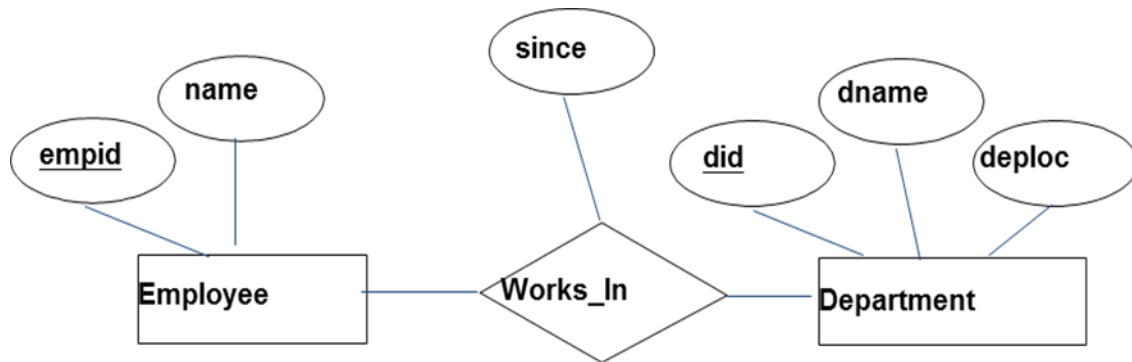
RELATIONSHIP TYPES, RELATIONSHIP SETS

Relationship:

- It is Association among two or more entities.
- When two or more entities are associated with each other, we have an instance of a *Relationship*.
- A relationship relates two or more distinct entities with a specific meaning.
- It is represented by a diamond.
- Relationships can have their own attributes

For Example EMPLOYEE john works in the PRODUCTION DEPT

- *Relationship WORKS-IN has Employee and Department as the participating entity sets*
- Whenever an attribute of one entity type refers to an entity (of the same or different entity type), we say that a relationship exists between the two entity types



Relationship Type:

- Relationship type is a group of relationships having same attributes.
- 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.

Relationship Set:

- It is Collection of similar relationships.
- Set of relationships over the same entity sets
- The current set of relationship instances represented in the database.
- The current *state of a relationship type*.
- Below figures shows employee set consists of 7 employee enties.

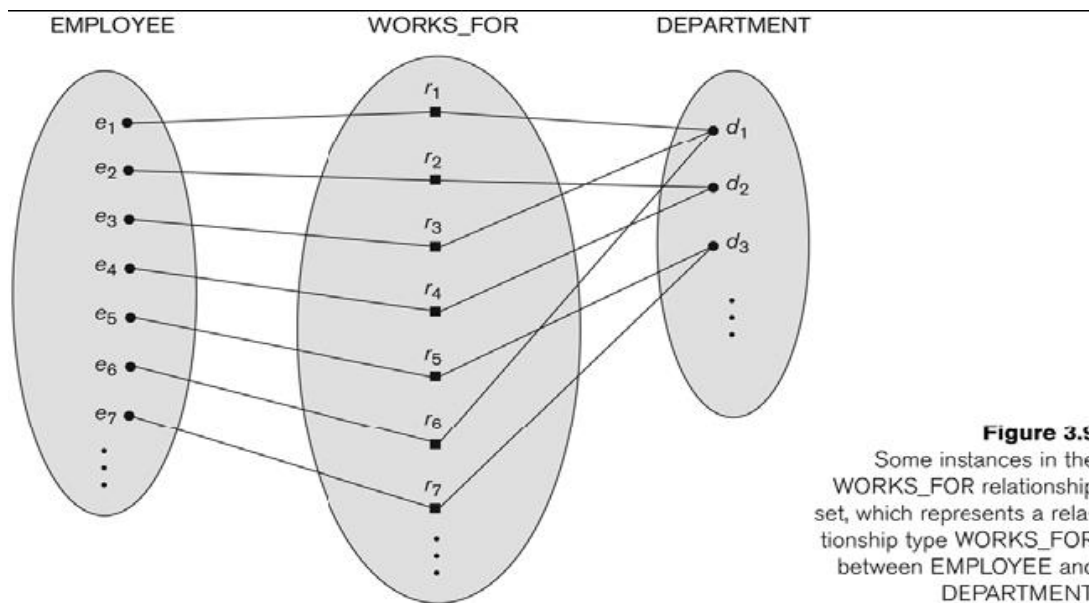


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship Degree

Degree of a Relationship Type.

- The degree of a relationship is the number of participating entities.
- It refers to number of entity sets that participate in a relationship set.
- The degree of WORKS-FOR relationship is two.
- A Relationship type of degree two is called **binary**, and a Relationship type of degree three is called **ternary**.
- 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.
- Binary relationships are very common and widely used.

Relationship Attributes

- A relationship type can have attributes describing properties of a relationship.
- FOR EXAMPLE Ramireddy works for CSE Department since 2011
- WOEKS-FOR relationship type have attributes like empid, deptid and since.

Constraints on Relationships

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
- For example if the company has a rule that each employee must work for exactly one department, then we would like to describe this constraint in the schema.

Two main types of relationship constraints

1. *Cardinality ratio*
2. *Participation.*

1 Cardinality Ratios for Binary Relationships

- The cardinality ratio for a binary relationship specifies the *maximum* number of entities of an entity can be associated with other entity via relationship set.

Following are the Cardinality Ratios for binary Relationships

1. One to One(1:1)
2. One to many(1:N)
3. Many to one(N:1)
4. Many to many (M:N)

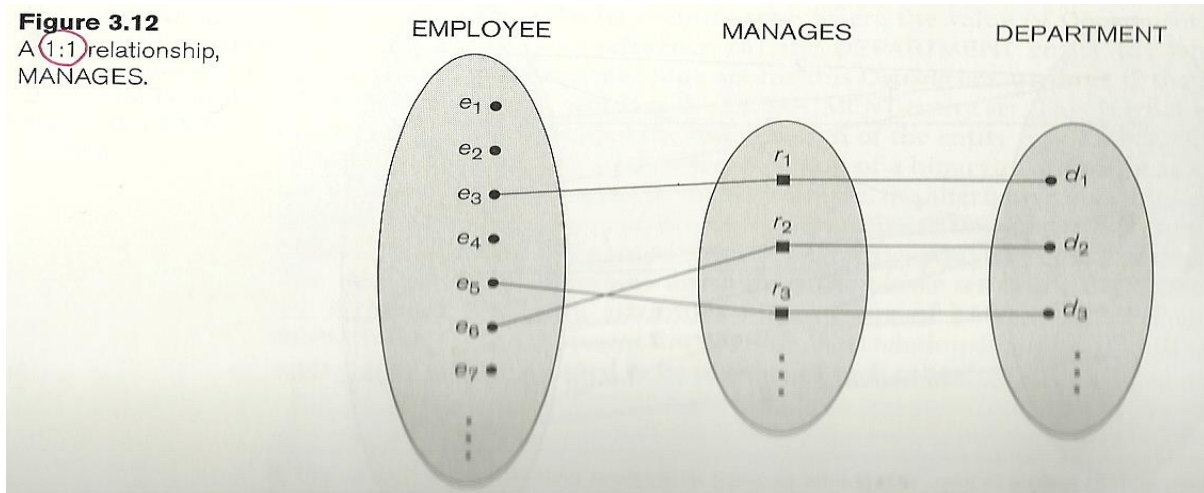
Consider binary relationship set R between entity sets A and B

One to One (1:1):

An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

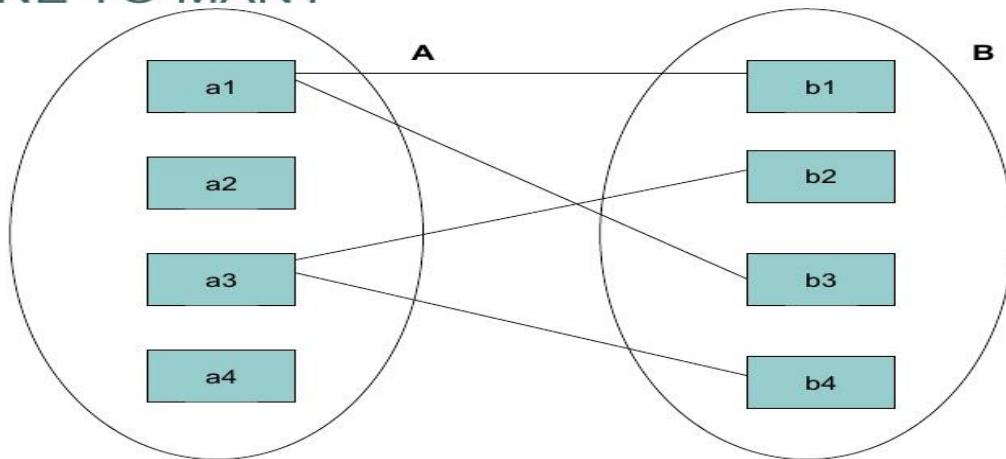
An example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department. This represents the mini-world constraints that at any point in time-an employee can manage only one department and a department has only one manager.

Figure 3.12
A 1:1 relationship,
MANAGES.



One to Many: An entity in A is associated with many entities in B and an entity in B is associated with at most one entity in A.

ONE TO MANY



Many to One: An entity in A is associated with at most one entity in B, an entity in B is associated with many entities in A.

Ex: an *employee* works in a single *department* but a *department* consists of many *employees*.

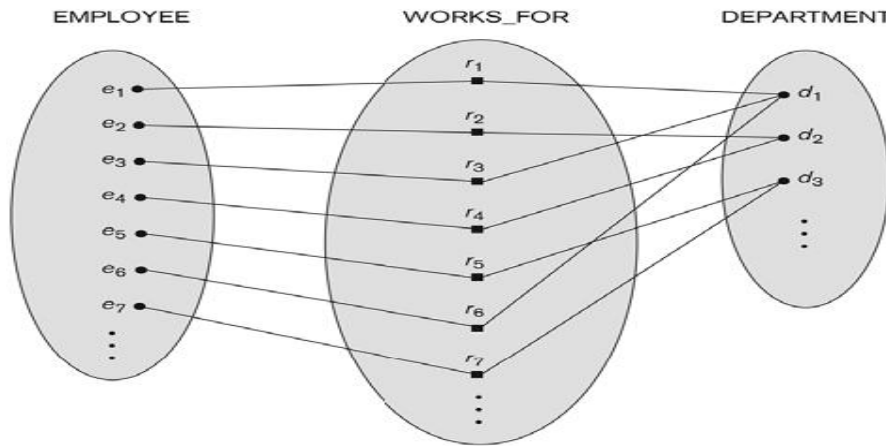


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many to Many: An entity in A is associated with many entities in B, and an entity in B is associated with many entities in A.

Ex: The relationship type WORKS_ON is of cardinality ratio M:N, because the mini-world rule is that an employee can work on several projects and a project can have several employees.

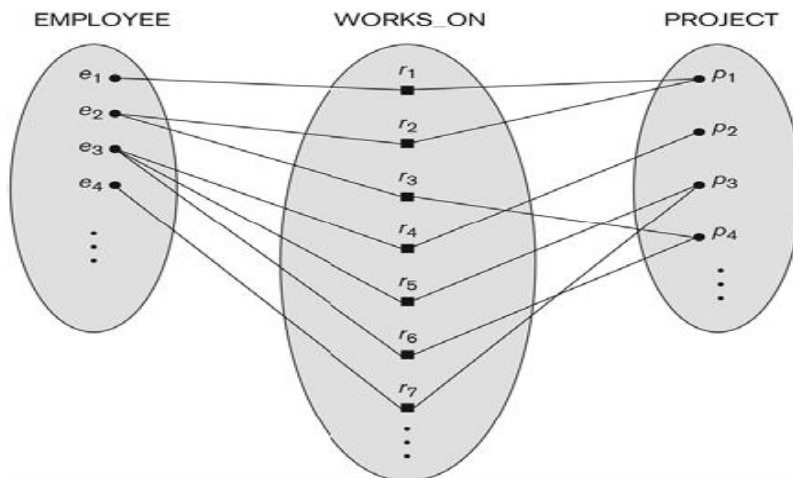


Figure 3.13
An M:N relationship, WORKS_ON.

2. Participation Constraints

This constraint specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the minimum cardinality constraint.

There are two types of participation constraints-

- 1 total participation
2. Partial participation

1. Total participation:

Total participation means every entity in an entity type must participated in a relationship set.

If a company policy states that *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. Total participation is also called existence dependency.

2. Partial participation:

Partial participation means only some of the entities of an entity type can be participated in a relationship set

We do not expect every employee to manage 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.

WEAK ENTITY TYPES

- Entity types that do not have key attributes of their own are called weak entity types.
- In contrast, regular entity types that do have a key attribute are also called strong 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

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

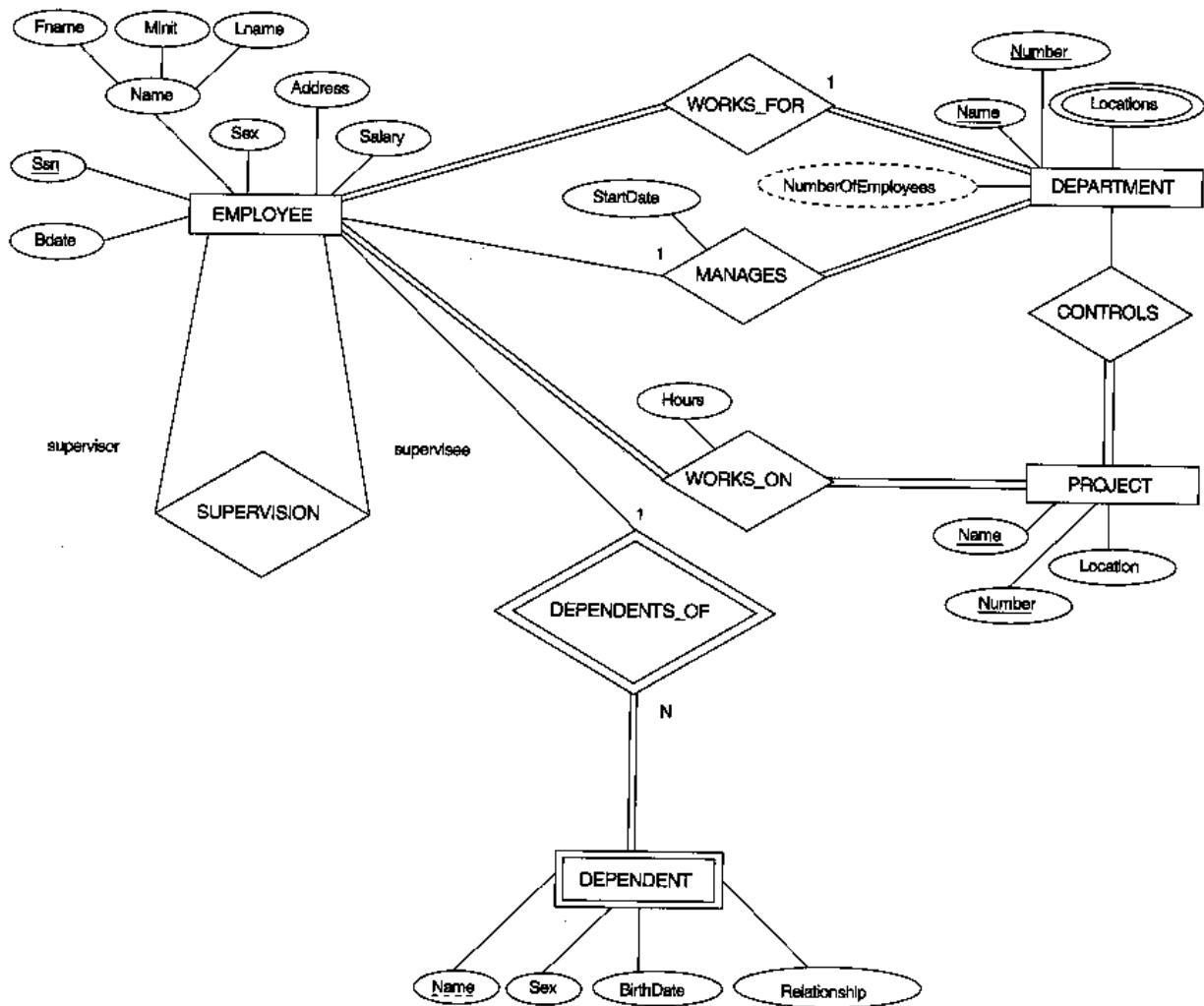


Figure 3.2 ER schema diagram for the COMPANY database.

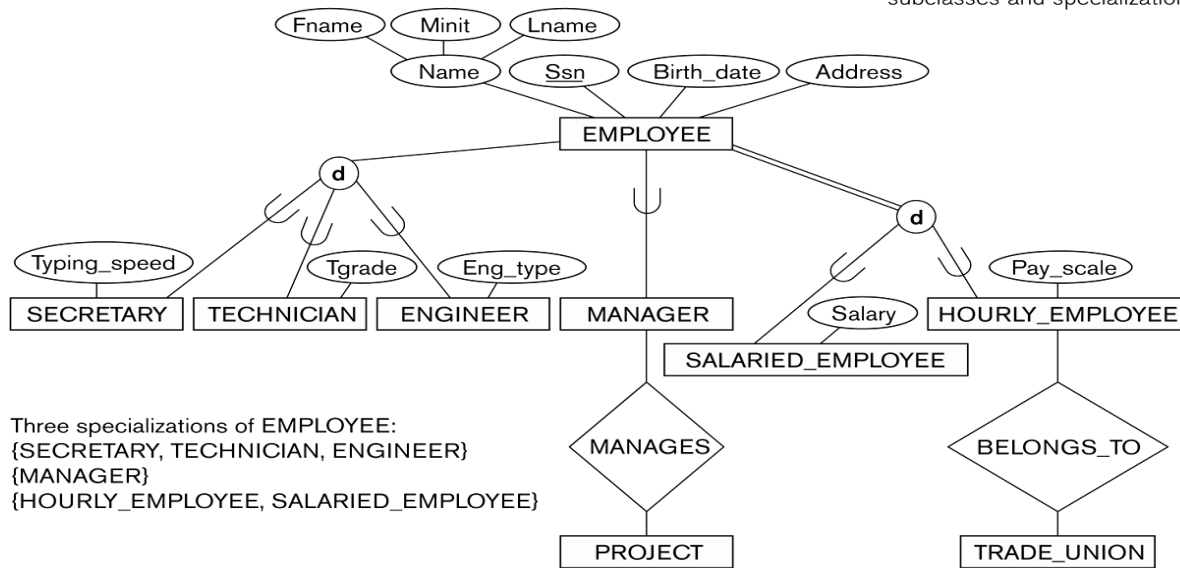
- The ER model is generally sufficient for "traditional" database applications.
- But more recent applications of DB technology (e.g., CAD/CAM, telecommunication, images/graphics, multimedia, data mining/warehousing, geographic info systems) cry out for a richer model.
- The EER (Enhanced ER) model includes all the modeling concepts of the ER model ,In addition, it includes the concepts of subclass and superclass and the related concepts of specialization and generalization.

- Another concept included in the EER model is that of a category or union type which is used to represent a collection of objects that is the *union* of objects of different entity types

SUBCLASSES, SUPERCLASSES AND INHERITANCE

- An entity type is used to represent both a *type of entity* and the *entity set* or *collection of entities of that type* that exist in the database.
- For example, the entity type EMPLOYEE describes the type (that is, the attributes and relationships) of each employee entity, and also refers to the current set of EMPLOYEE entities in the COMPANY database.
- In many cases an entity type has numerous sub groupings of its entities that are meaningful and need to be represented explicitly because of their significance to the database application. For example, the entities that are members of the EMPLOYEE entity type may be grouped further into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on.
- The set of entities in each of the latter groupings is a subset of the entities that belong to the EMPLOYEE entity set, meaning that every entity that is a member of one of these sub groupings is also an employee.
- We call each of these sub groupings a subclass of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the super class for each of these subclasses
- In our previous example, EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are two class/subclass relationships.

Figure 4.1
EER diagram notation to represent subclasses and specialization.



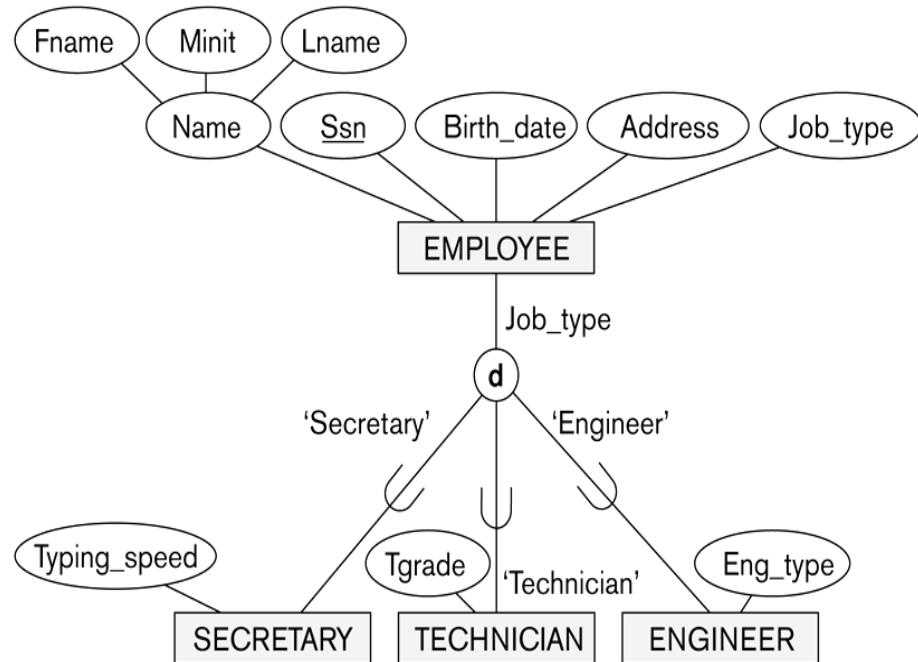
- An important concept associated with subclasses is that of type inheritance. Recall that the *type* of an entity is defined by the attributes it possesses and the relationship types in which it participates.
- An entity that is a member of a subclass inherits all the attributes of the entity as a member of the super class.
- The entity also inherits all the relationships in which the super class participates.
- Notice that a subclass, with its own specific (or local) attributes and relationships together with all the attributes and relationships it inherits from the super class, can be considered an *entity type*.

Specialization

- Specialization is the process of defining a *set of subclasses* of an entity type;
- That entity type is called the super class of the specialization.
- The set of subclasses that form a specialization is defined on the basis of some distinguishing characteristic of the entities in the super class.

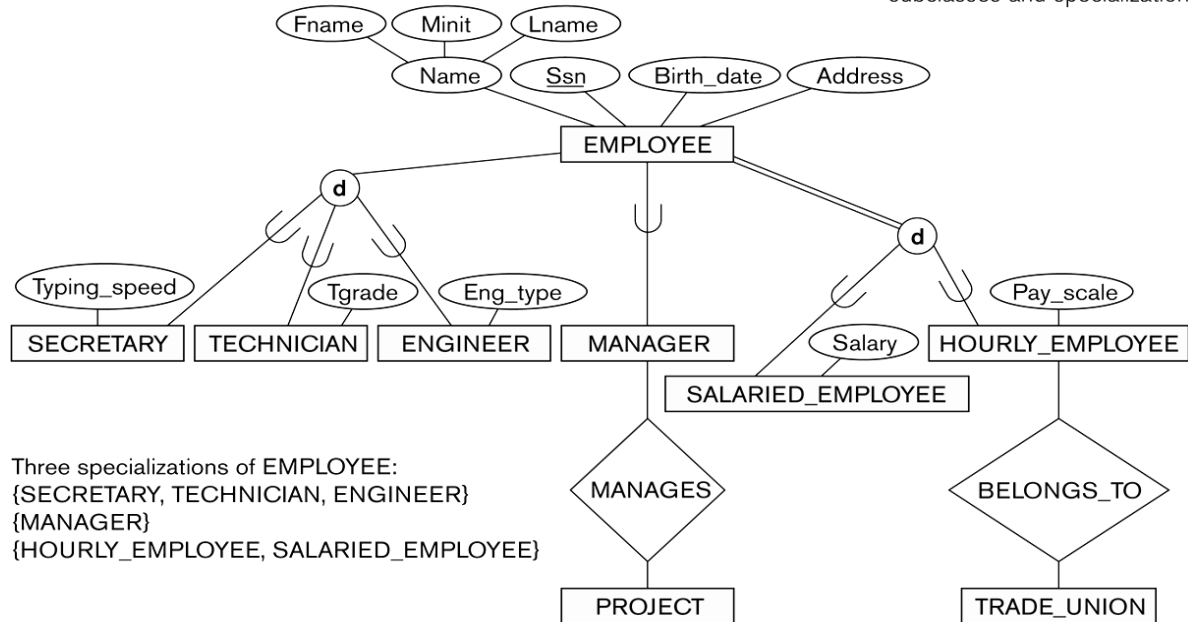
Figure 4.4

EER diagram notation for an attribute-defined specialization on Job_type.



- For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the super class EMPLOYEE that distinguishes among employee entities based on the *job type* of each employee entity.
- We may have several specializations of the same entity type based on different distinguishing characteristics.
- For example, another specialization of the EMPLOYEE entity type may yield the set of subclasses {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}; this specialization distinguishes among employees based on the *method of pay*.

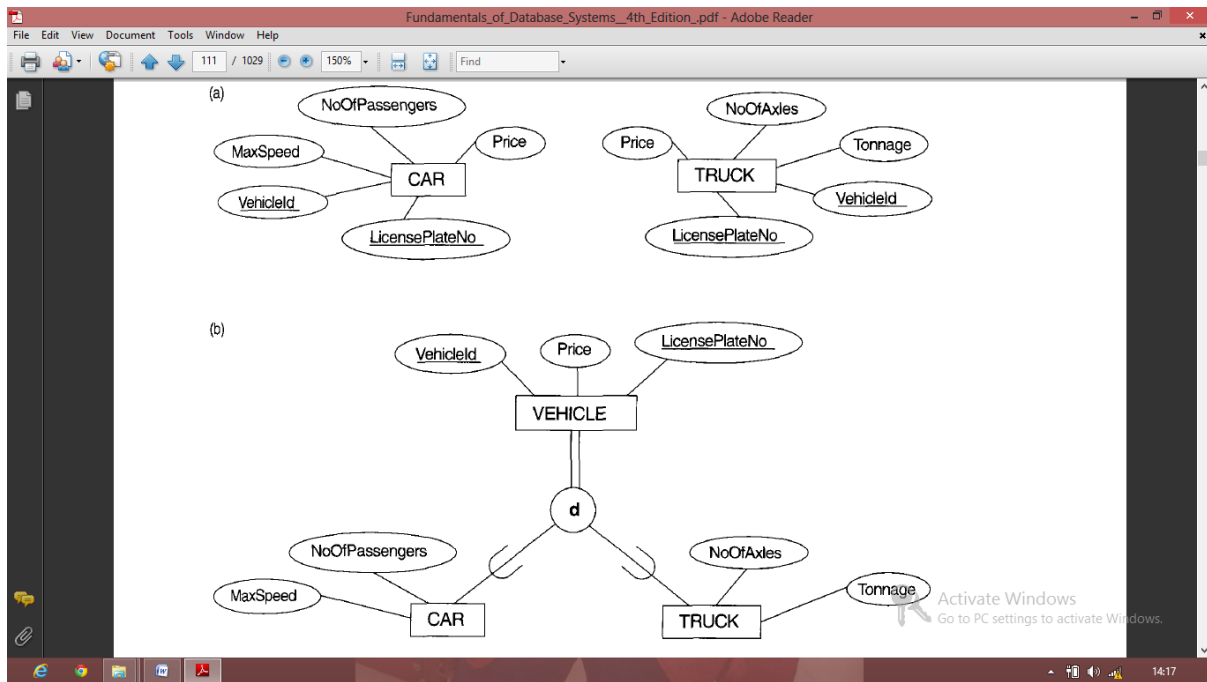
Figure 4.1
EER diagram notation to represent subclasses and specialization.



- Above figure shows how we represent a specialization diagrammatically in an EER diagram.
- The subclasses that define a specialization are attached by lines to a circle that represents the specialization, which is connected to the super class.
- The *subset symbol* on each line connecting a subclass to the circle indicates the direction of the super class/subclass relationship.
- Attributes that apply only to entities of a particular subclass—such as Typing Speed of SECRETARY—are attached to the rectangle representing that subclass. These are called specific attributes (or local attributes) of the subclass.
- Similarly, a subclass can participate in specific relationship types, such as the HOURLY_EMPLOYEE subclass participating in the BELONGS_TO relationship in above figure.

Generalization

- The term generalization to refer to the process of defining a generalized entity type from the given entity types.
- **Generalization** a *reverse process* of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single super class of which the original entity types are special subclasses.
- For example, consider the entity types CAR and TRUCK shown in below Figure. Because they have several common attributes, they can be generalized into the entity type VEHICLE, as shown in below Figure. Both CAR and TRUCK are now subclasses of the generalized super class VEHICLE.
- Notice that the generalization process can be viewed as being functionally the inverse of the specialization process.
- Hence, in below Figure we can view {CAR, TRUCK} as a specialization of VEHICLE, rather than viewing VEHICLE as a generalization of CAR and TRUCK.
- Similarly, in previous figure we can view EMPLOYEE as a generalization of SECRETARY, TECHNICIAN, and ENGINEER.
- A diagrammatic notation to distinguish between generalization and specialization is used in some design methodologies. An arrow pointing to the generalized super class represents a generalization, whereas arrows pointing to the specialized subclasses represent a specialization



E-R Diagram for Company Database

An ER diagram for a BANK database schema.

