PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIT-2

Subject Name: **Database Management Systems**
Subject Code: **20CS3402**

SYLLABUS:

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, ER Diagrams, Naming Conventions and Design Issues, Relationship Types of Degree Higher Than Two. RelationalDatabase Design Using ER-to-Relational Mapping.

High-Level Conceptual Data Models for Database Design:

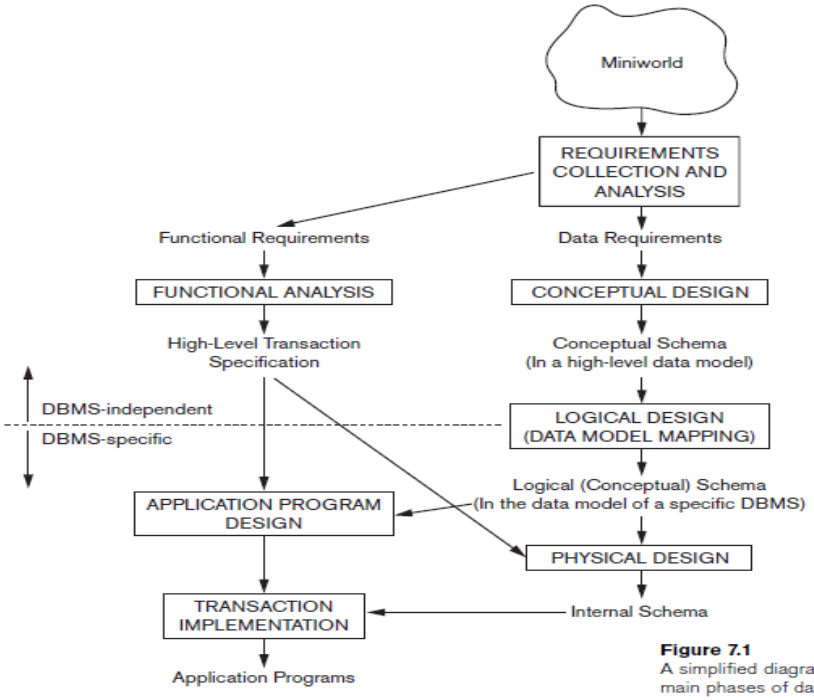


Figure 7.1
A simplified diagram to illustrate the main phases of database design.

Requirements collection and analysis:

- During this step, the database designers interview prospective database users to understand and document their data requirements.
- These requirements should be specified in as detailed and complete a form as possible.
- In parallel with specifying the data requirements, it is useful to specify the known functional requirements of the application.

- These consist of the user defined operations (or transactions) that will be applied to the database, including both retrievals and updates.

Conceptual Design

- 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 using the concepts provided by the high-level data model.

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—so 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 mappining.**

Physical Design

- The last step is the **physical design** phase, during which the internal storage structures, file organizations, indexes, access paths, and physical design parameters 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.

A Sample Database Application

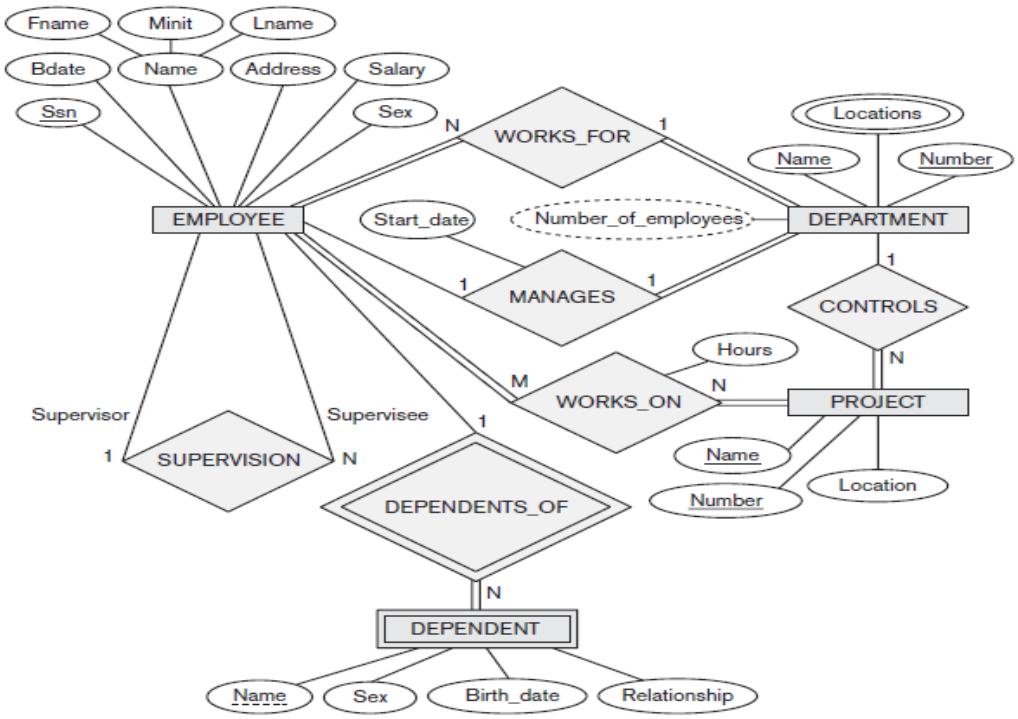
- The COMPANY database keeps track of a company's employees, departments, and projects.
- Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the Mini world—the part of the company that will be represented in the database.

➢ 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**.

➢ We store each **employee's name, Social Security number, address, salary, 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.

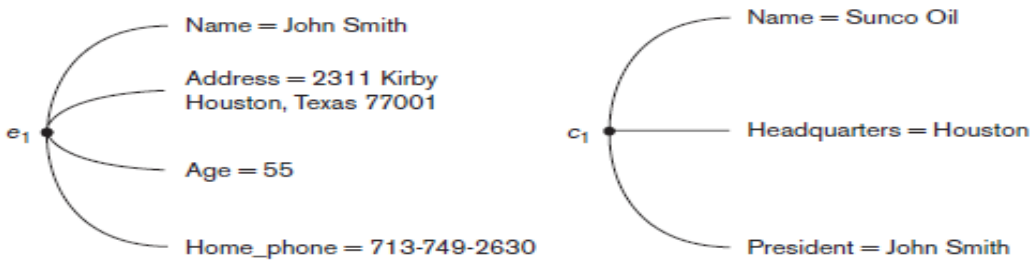
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's **first name, Gender, birth date, and relationship to the employee.**



Entity Types, Entity Sets, Attributes and Keys

Entities and Attributes:

- The basic object that the ER model represents is 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)
- Each entity has attributes—the particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.



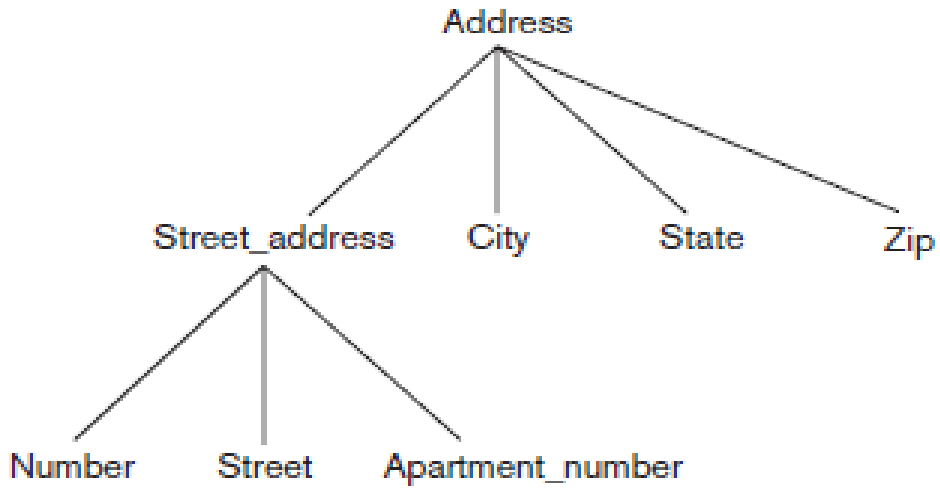
Several types of attributes occur in the ER model:

- simple versus composite,
- Single valued versus multivalued, and
- stored versus derived.

- Null Values.
- Complex attributes.

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 Street_address, City, State, and Zip.



Attributes that are not divisible are called **simple** or **atomic attributes**.
For example, Age.

Single-Valued versus Multivalued Attributes:

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.

In some cases an attribute can have a set of values for the same entity—for instance, a Colors attribute for a car, or a College_degrees attribute for a person. Such attributes are called **multivalued.**

Stored versus Derived Attributes:

For example, the Age and Birth_date attributes of a person. For a particular person entity, the value of Age can be determined from the current date and the value of that person's Birth_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute.

NULL Values:

In some cases, a particular entity may not have an applicable value for an attribute. For example, the Apartment_number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes. NULL can also be used if we do not know the value of an attribute for a particular entity.

Complex Attributes

If an attribute for an entity, is built using composite and multivalued attributes, then these attributes are called complex attributes. For example, a person can have more than one residence and each residence can have multiple phones, an address phone for a person entity can be specified as:

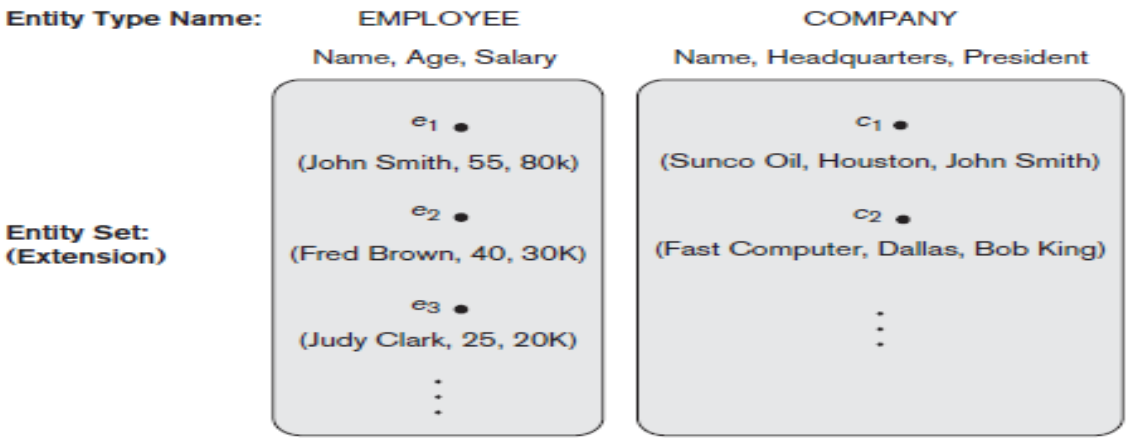
{ Address phone (phone {(Area Code, Phone Number)},

Address (Sector Address (Sector Number, House Number), City, State, Pin)) }

Here {} are used to enclose multivalued attributes and () are used to enclose composite attributes with comma separating individual attributes .

Entity Types and Entity Sets:

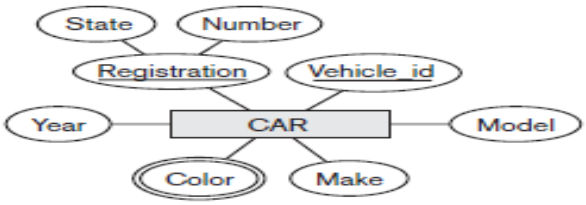
- An **entity type** defines a collection (or set) of entities that have the same attributes. Each entity type in the database is described by its name and attributes.
- **For example**, a company employing hundreds of employees may want to store similar information concerning each of the employees.
- 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.
- **For example**, EMPLOYEE refers to both a type of entity as well as the current set of all employee entities in the database.



Key Attributes of an Entity Type:

- An important constraint on the entities of an entity type is the **key** or **uniqueness constraint** on attributes.

(a)



(b)

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

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Value Sets (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.

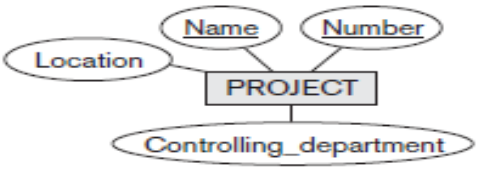
Initial Conceptual Design of the COMPANY Database:

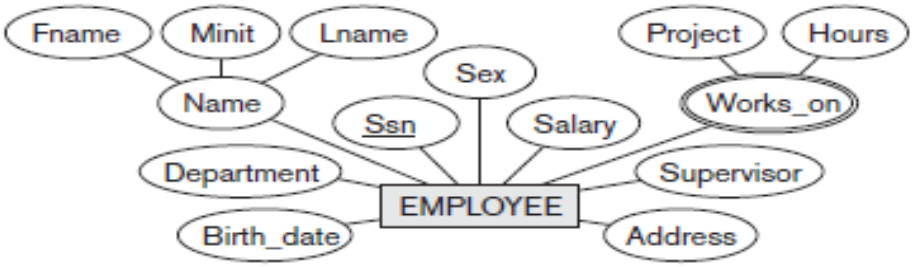
1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.

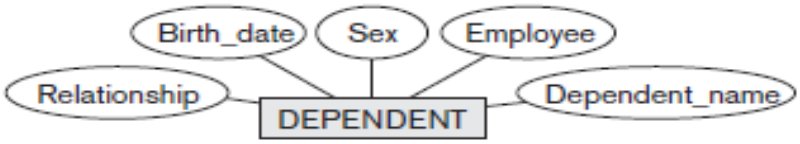
2. An entity type PROJECT with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.

3. An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial, Last_name—or of Address.

4. An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).







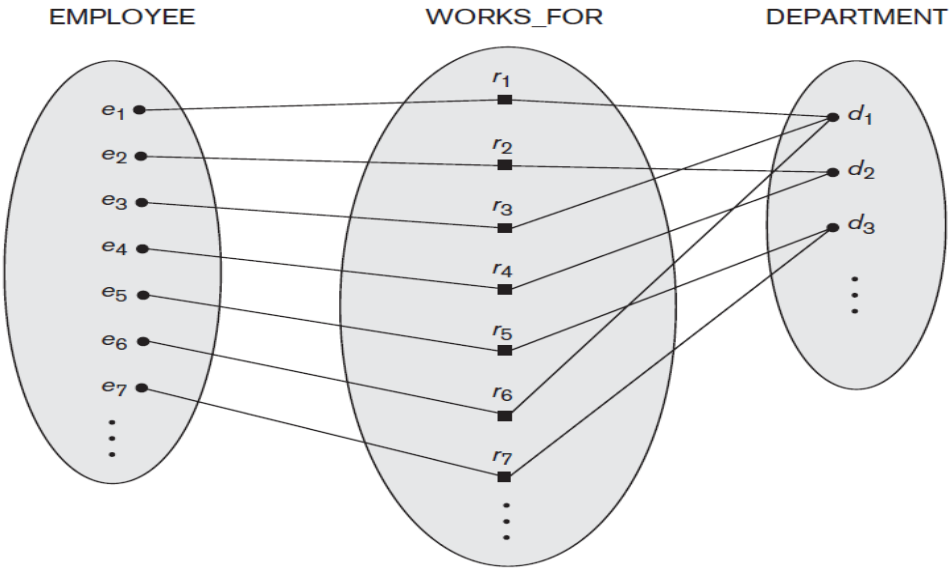
Relationship Types, Relationship Sets, Roles, and Structural Constraints:

Relationship Types, Sets, and Instances:

- A **relationship type** *R* among *n* entity types *E*1, *E*2, ..., *En* defines a set of associations**. (OR)**

  Association among two or more entities.

- A **relationship set**—among entities from these entity types.
- Entity types and entity sets, a relationship type and its corresponding relationship set are customarily referred to by the *same name*, the relationship set *R* is a set of **relationship instances** *ri*.

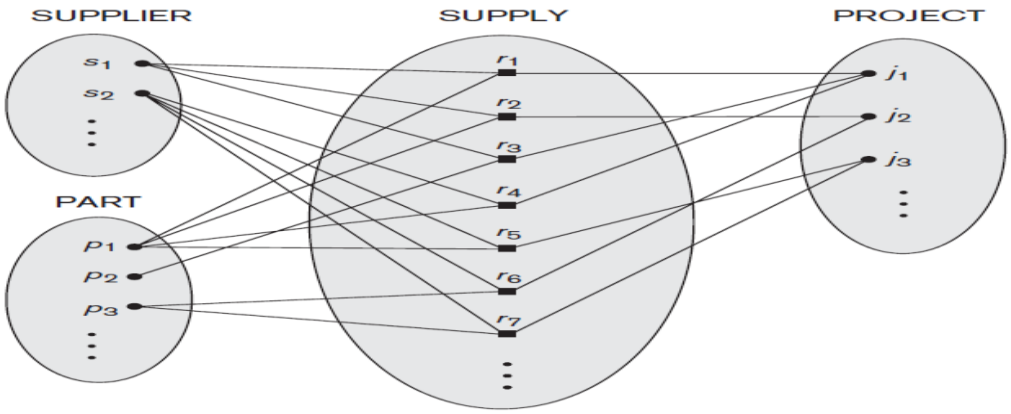


Relationship Degree, Role Names, and Recursive Relationships:

Degree of a Relationship Type:

- The degree of a relationship type is the number of participating entity types. Hence, the WORKS_FOR relationship is of degree two.

- A relationship type of degree two is called binary, and one of degree three is called ternary.

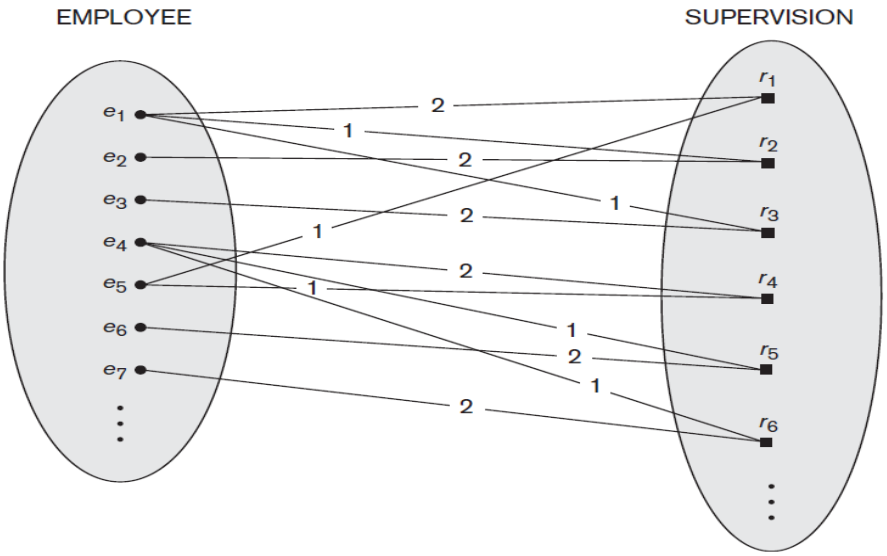


Role Names and Recursive Relationships:

- The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means.

  For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.

- 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**.



Constraints on Binary Relationship Types:

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
- Two main types of binary relationship constraints: **cardinality ratio** and **participation**.

Cardinality Ratios for Binary Relationships:

- The **cardinality ratio** for a binary relationship specifies the *maximum* number of

relationship instances that an entity can participate in.

- **For example**, in the WORKS_FOR binary relationship type, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

Types of Cardinality :

There can be 4 types of cardinality –

1) One-to-one (1:1)- When one entity in each entity set takes part at most once in the relationship, the cardinality is one-to-one.

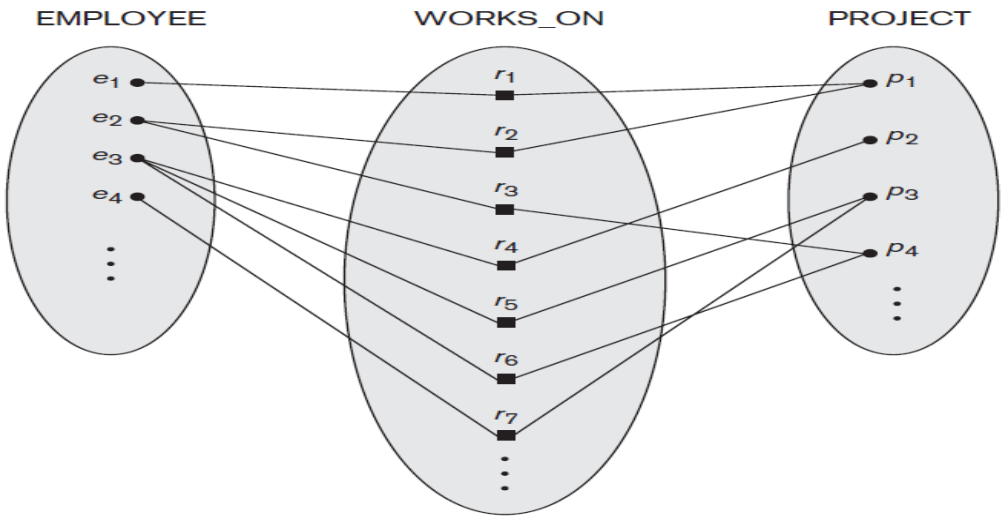
2) One-to-many (1: N) – If entities in the first entity set take part in the relationship set at most once and entities in the second entity set take part many times (at least twice), the cardinality is said to be one-to-many.

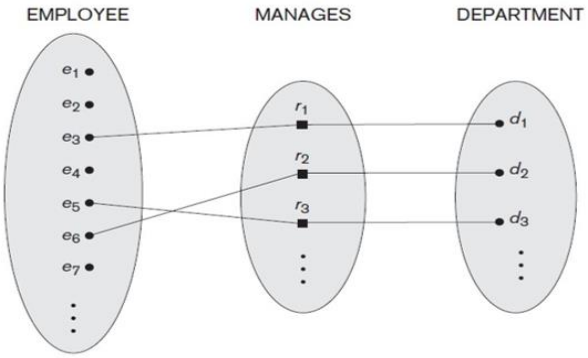
3) Many-to-one (N:1) – If entities in the first entity set take part in the relationship set many times (at least twice), while entities in the second entity set take part at most once, the cardinality is said to be many-to-one.

4) Many-to-many (N: N) – The cardinality is said to be many to many if entities in both the entity sets take part many times (at least twice) in the relationship set.

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, and is sometimes called the minimum cardinality constraint.
- There are two types of participation constraints—total and partial.
- The cardinality ratio and participation constraints, taken together, as the structural constraints of a relationship type.
- 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* .





Attributes of Relationship Types:

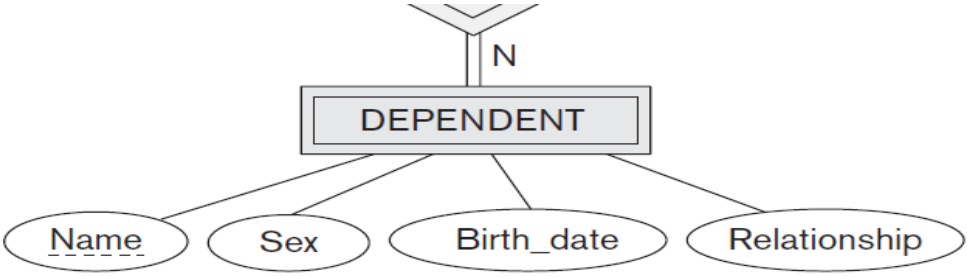
Relationship types can also have attributes, similar to those of entity types**.**

- For example, to record the number of hours per week that an **employee** works on a particular **project**, we can include an attribute Hours for the **WORKS_ON** relationship type.
- Another example is to include the date on which a manager started managing a **department** via an attribute Start_date for the **MANAGES** relationship type.

Weak Entity Types:

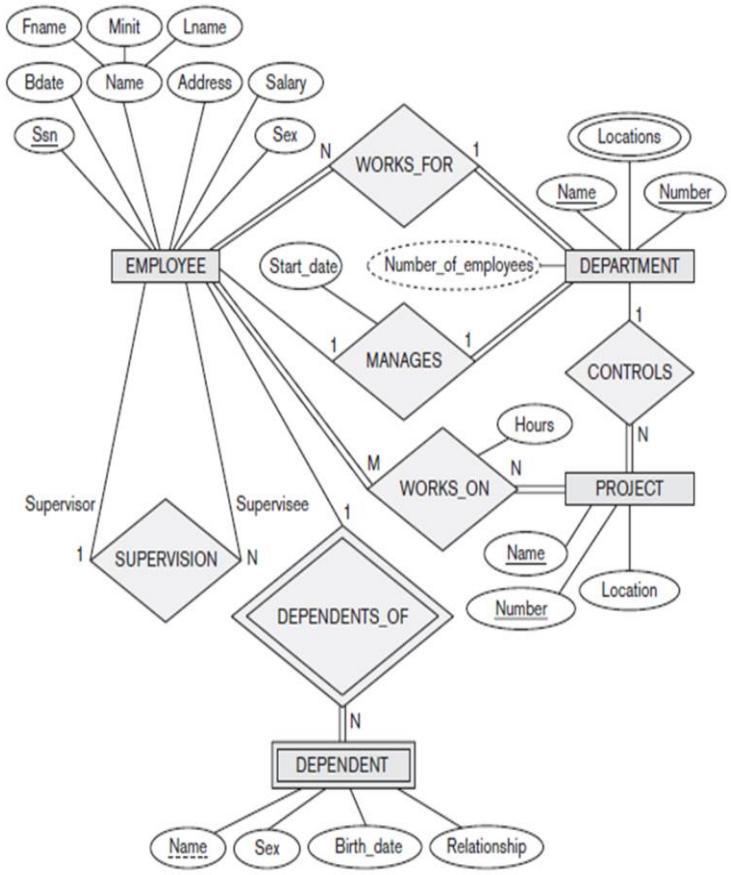
- Entity types that do not have key attributes of their own are called **weak entityTypes.**
- Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.
- Entity type the **identifying** or **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.
- A weak entity type always has a *total participation constraint* (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.
- 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.
- 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,Birth_date, Sex, and Relationship.
- 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, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines . The partial key attribute is

underlined with a dashed or dotted line.



Refining the ER Design for the COMPANY Database:

- In our example, we specify the following relationship types:
- MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation. The attribute Start_date is assigned to this relationship type.
- WORKS_FOR, a 1:N 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.
- 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, determined to be an M:N 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
- Proper Naming of Schema Constructs
- Design Choices for ER Conceptual Design
- Alternative Notations for ER Diagrams

Summary of Notation for ER Diagrams:

- The participation of entity types in relationship types by displaying their sets or extensions—the individual entity instances in an entity set and the individual relationship instances in a relationship set.
- The COMPANY **ER database schema** as an **ER diagram.**

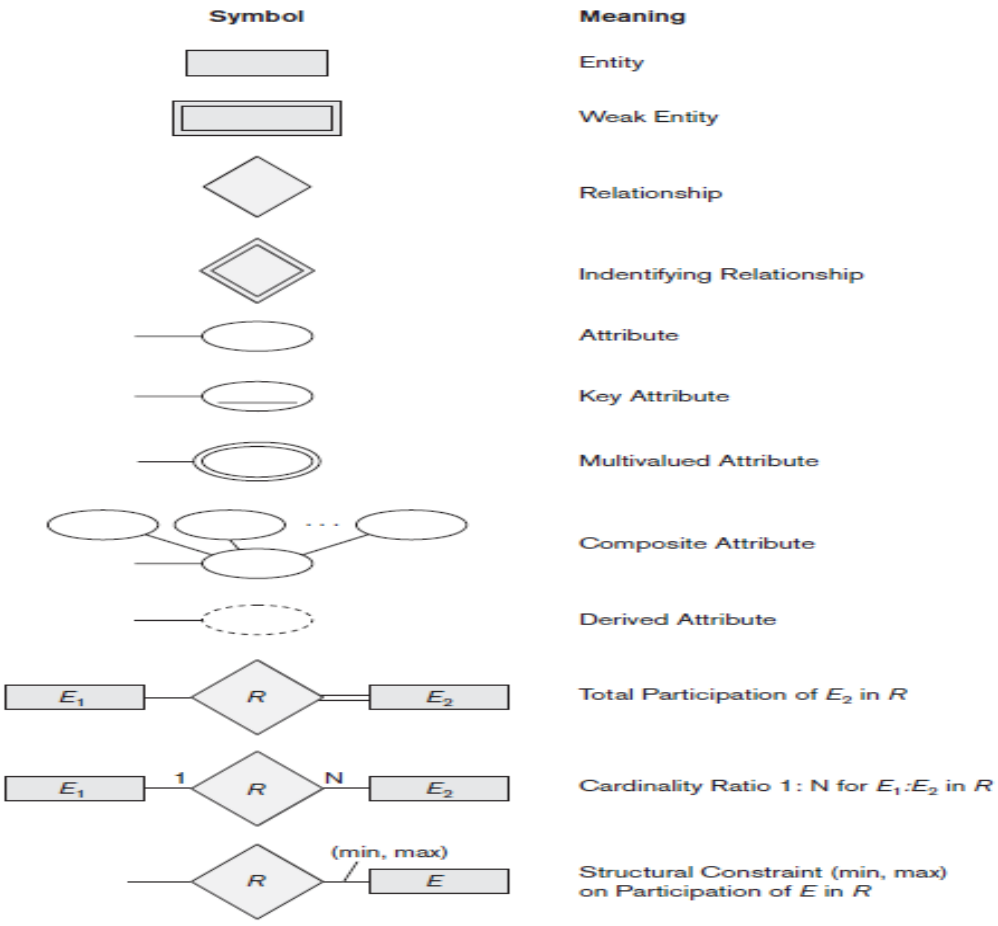
  ---Entity types and Relation types

  --Weak entity and Identifying relationship type

  --Cardinality ration of binary relationship

  --Participation Constraint

--Role names



Proper Naming of Schema Constructs:

- When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straightforward.

  -> entity type and relationship type names are uppercase letters

  -> attribute names have their initial letter capitalized

  -> role names are lowercase letters

- Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

Design Choices for ER Conceptual Design:

- It is occasionally difficult to decide whether a particular concept in the mini world should be modeled as an entity type, an attribute, or a relationship type.

  * A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type.

  *An attribute that exists in several entity types may be elevated or promoted to an

independent entity type.

For example: suppose that several entity types in a UNIVERSITY database, such as **STUDENT, INSTRUCTOR**, and **COURSE**, each has an attribute Department in the initial design; the designer may then choose to create an entity type DEPARTMENT with a single attribute Dept_name and relate it to the three entity types (STUDENT, INSTRUCTOR, and COURSE) via appropriate relationships.

*An inverse refinement to the previous case may be applied—for example, if an entity type DEPARTMENT exists in the initial design with a single attribute Dept_name and is related to only one other entity type, STUDENT. In this case, DEPARTMENT may be reduced or demoted to an attribute of STUDENT.

Alternative Notations for ER Diagrams:

- There are many alternative diagrammatic notations for displaying ER diagrams.
- The Unified Modeling Language (UML) notation for class diagrams, which has been proposed as a standard for conceptual object modeling.
- one alternative ER notation for specifying structural constraints on relationships, which replaces the cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints.
- This notation involves associating a pair of integer numbers (min, max) with each *participation* of an entity type *E* in a relationship type *R*.
- In this method, min = 0 implies partial participation, whereas min > 0 implies total participation.

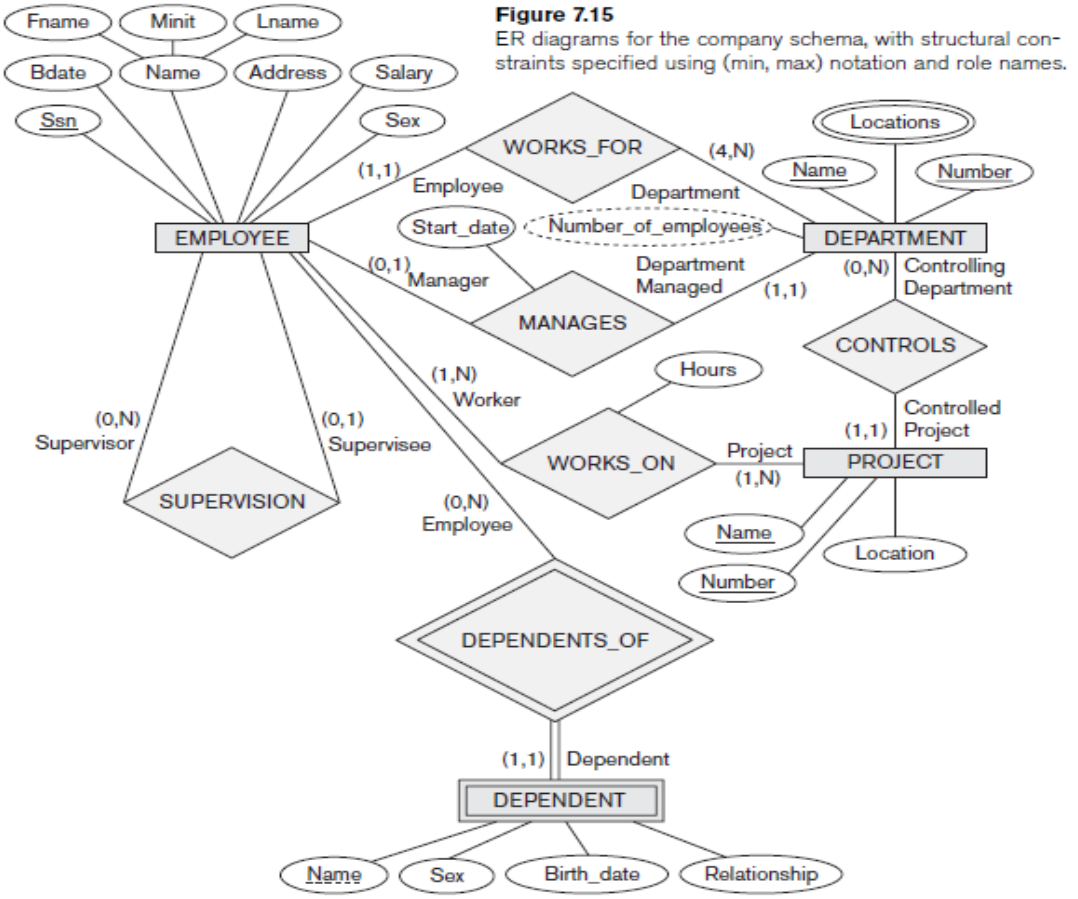
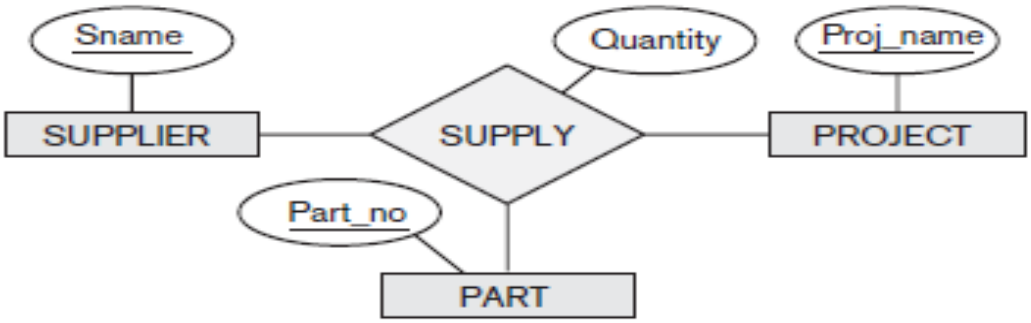


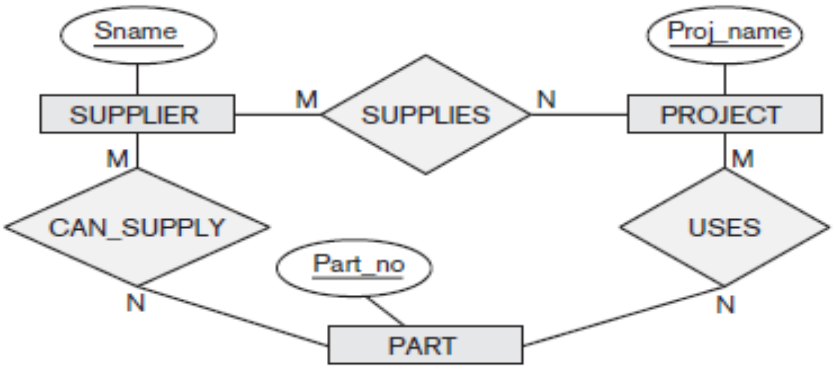
Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Relationship Types of Degree Higher Than Two:

The degree of a relationship type as the number of participating entity types and called a relationship type of degree two binary and a relationship type of degree three ternary.

- Choosing between Binary and Ternary(or Higher-Degree) Relationships.
- Constraints on Ternary (or Higher-Degree) Relationships.
- **Choosing between Binary and Ternary(or Higher-Degree) Relationships**:





ER diagram for three binary relationship types CAN_SUPPLY, USES, and SUPPLIES. CAN_SUPPLY between SUPPLIER and PART, includes an instance (s, p) whenever supplier s can supply part p (to any project). USES between PROJECT and PART, includes an instance (j , p) whenever project j uses part p. SUPPLIES between SUPPLIER and PROJECT, includes an instance (s, j) whenever supplier s supplies some part to project j.

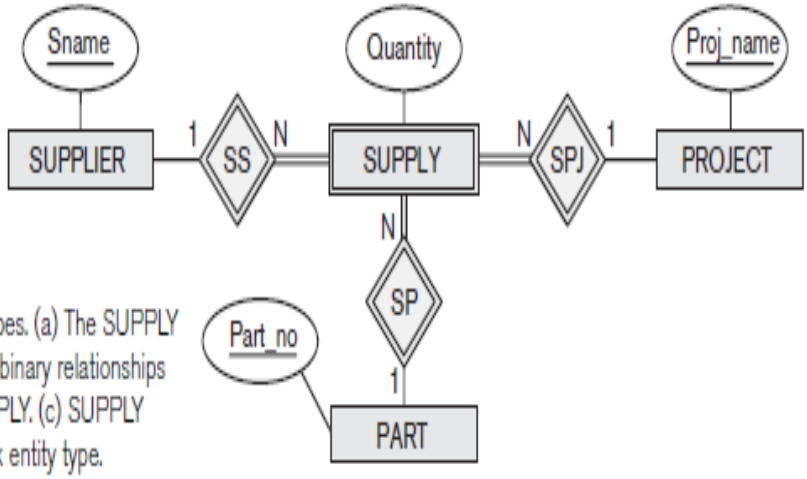


Figure 7.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Some database design tools are based on variations of the ER model that permit only binary relationships. In this case, a ternary relationship such as SUPPLY must be represented as a weak entity type, with no partial key and with three identifying relationships.

Constraints on Ternary (or Higher-Degree) Relationships:

- There are two notations for specifying structural constraints on *n*-ary relationships, and they specify different constraints.
- They should thus both be used if it is important to fully specify the structural constraints on a ternary or higher-degree relationship.
- The first notation is based on the cardinality ratio notation of binary Relationships.
- The second notation is based on the (min, max) notation for binary relationships.
- A (min, max) on a participation here specifies that each entity is related to at least min and

at most max relationship instances in the relationship set.

Relational Database Design Using ER-to-Relational Mapping:

Step 1: Mapping of Regular Entity types (Strong Entities).

Step 2: Mapping of Weak Entity types.

Step 3: Mapping of Binary 1:1 Relationship types.

Step 4: Mapping of Binary 1:N Relationship types.

Step 5: Mapping of Binary M:N Relationship types.

Step 6: Mapping of Multivalued Attributes.

Step 7: Mapping of N-ary Relationship types.

Step 1: Mapping of Regular Entity Types.

For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown. The ER conceptual schema diagram for the COMPANY database.

Step 2: Mapping of Weak Entity

Types For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R. Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Example: Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN). The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

Step 3: Mapping of Binary 1:1 Relation 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:

Foreign Key approach: Choose one of the relations-say S-and include 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.

Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

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

For each regular binary 1:N relationship type R, identify the relation S that represent 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. Include any simple attributes of the 1:N relation type as attributes of S.

Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure. For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

Step 5: Mapping of Binary M:N Relationship Types.

For each regular binary M:N relationship type R, 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. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema. The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively. Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

Step 6: Mapping of Multivalued attributes.

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 of relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Example: The relation DEPT_LOCATIONS is created. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}.

Step 7: Mapping of N-ary Relationship Types.

For each n-ary relationship type R, where n>2, create a new relationship S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.