Chapter Three
Conceptual Database Design: E-R modeling

Conceptual modeling is an important phase in designing a successful database application. Generally, the term **database application** refers to a particular database—for example, a BANK database that keeps track of customer accounts—and the associated *programs* that implement the database queries and updates—for example, programs that implement database updates corresponding to customers making deposits and withdrawals. These programs often provide user-friendly graphical user interfaces (GUIs) utilizing forms and menus.

We will present the modeling concepts of the **Entity-Relationship (ER) model,** which is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts. The ER model describes data as entities, relationships, and attributes.

3.1. The Entity Relationship (E-R) Model

- ✓ An entity-relationship model (ERM) is a model that provides a high-level description of a conceptual data model. Data modeling provides a graphical notation for representing such data models in the form of entity-relationship diagrams (ERD).
- ✓ Entity-Relationship modeling is used to represent conceptual view of the database
- ✓ The whole purpose of ER modeling is to create an accurate reflection of the real world in a database. The ER model doesn't actually give us a database description. It gives us an intermediate step from which it is easy to define a database.
- ✓ The E-R data model is based on a perception of a real world that consists of a set of basic objects called *entities,* and of *relationships* among these objects. It was developed to facilitate database design by allowing the specification of an *enterprise schema,* which represents the overall logical structure of a database.
- ✓ The E-R data model is one of several semantic data models; the semantic aspect of the model lies in the attempt to represent the meaning of the data. The E-R model is extremely useful in mapping the meanings and interactions of real-world enterprises onto a conceptual scheme. Because of this utility, many database design tools draw on concepts from the E-R model.

- ✓ The main components of ER Modeling are:

- **Entities**
- **Attribute**
- **Relationship**

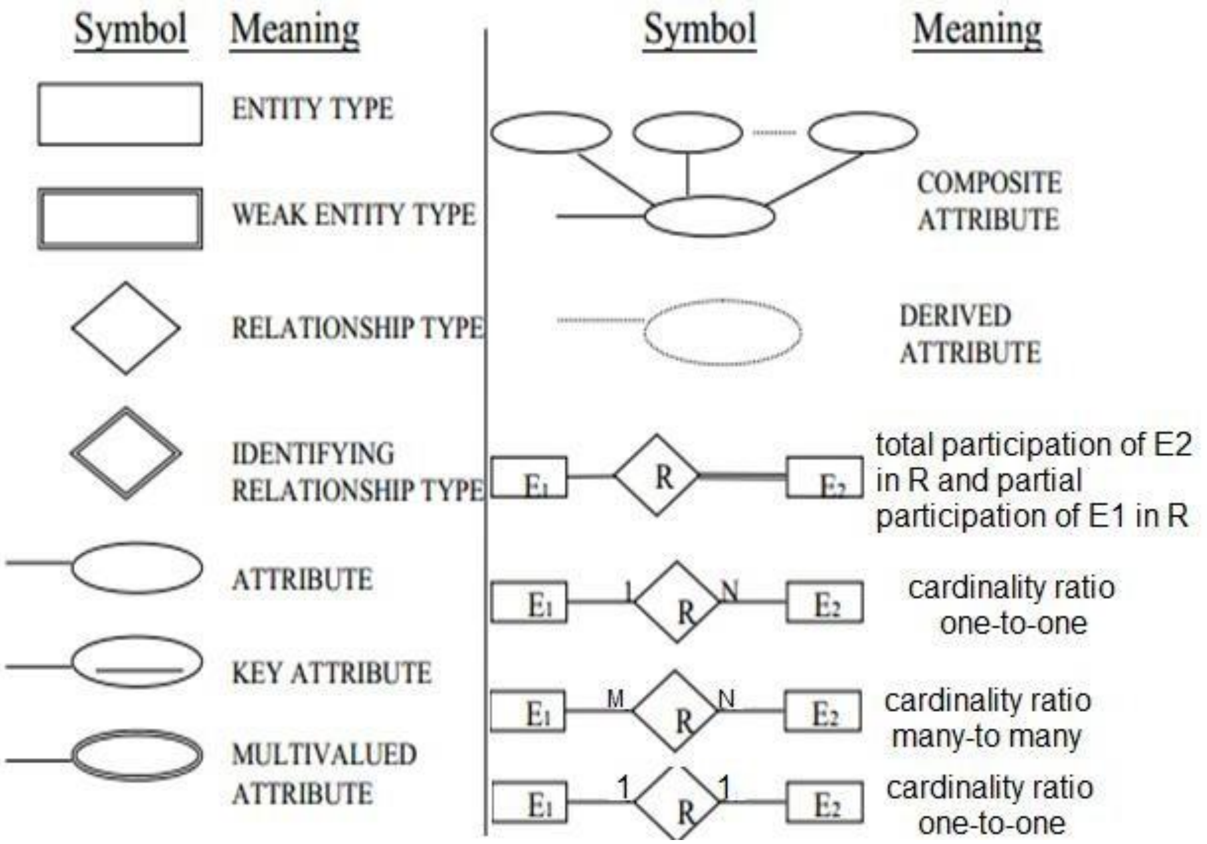
Before working on the conceptual design of the database, one has to know and answer the following basic questions.

- ✓ What are the entities and relationships in the enterprise?
- ✓ What information about these entities and relationships should we store in the database?
- ✓ What are the integrity constraints that hold? Constraints on each data with respect to update, retrieval and store.
- ✓ Represent this information pictorially in ER diagrams, then map ER diagram into a relational schema.

Steps to draw ER-diagram

- Identify Entities
    - Classify them as strong or weak
        - ***Diagramming entities***
- Identify the attributes for identified Entities
    - Decide types of attributes
        - ***Diagramming attributes***
- Establish relationships between Entity(s)
    - Identify the relationship type based on degree
    - Identify the relationship type based on cardinality ratio
    - Identify the relationship type based on participation constraint
        - ***Diagramming relationships***
- Draw final ER-Diagram

Summary of ER Diagram Notation


Symbol
Meaning
Symbol
Meaning
ENTITY TYPE
WEAK ENTITY TYPE
RELATIONSHIP TYPE
IDENTIFYING RELATIONSHIP TYPE
ATTRIBUTE
KEY ATTRIBUTE
MULTIVALUED ATTRIBUTE
COMPOSITE ATTRIBUTE
DERIVED ATTRIBUTE
E1
R
E2
total participation of E2 in R and partial participation of E1 in R
E1
1
R
N
E2
cardinality ratio one-to-one
E1
M
R
N
E2
cardinality ratio many-to many
E1
1
R
1
E2
cardinality ratio one-to-one

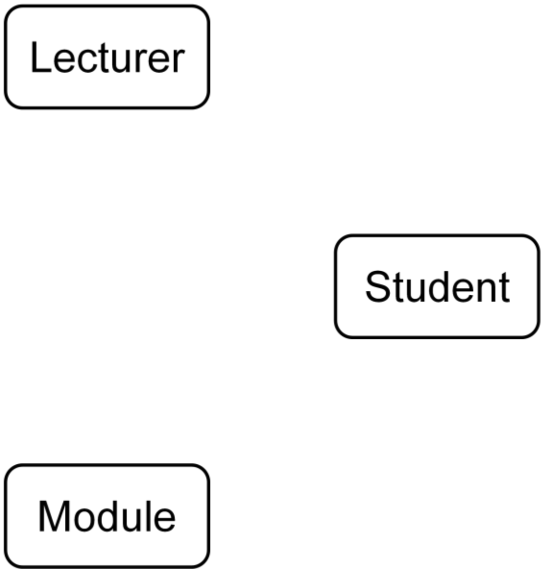
Example

Requirements of the university (oversimplified for illustrative purposes)

- In a University database we might have entities for Students, Modules and Lecturers. Students might have attributes such as their ID, Name, and Course whereas Lecturers might have attributes such as LID, Name, Rank and Modules might have attributes such as ModuleNumber, ModuleName, CrHr and could have relationships (studies) with Student and Modules and (*tutor*) with Lecturers.

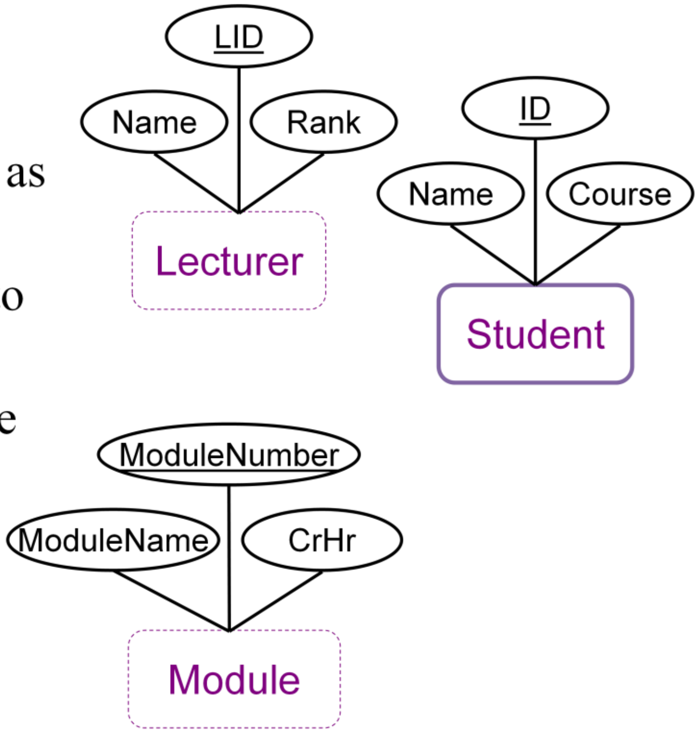
Diagramming Entities

- In an E/R Diagram, an entity is usually drawn as a box with rounded corners
- The box is labelled with the name of the class of objects represented by that entity



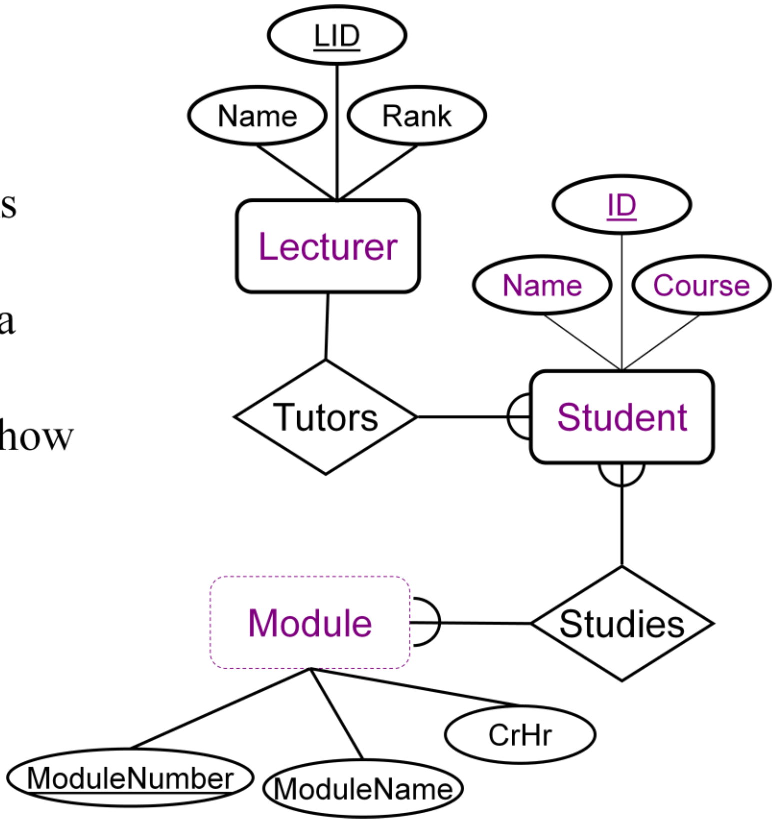
Diagramming Attributes

- In an E/R Diagram attributes may be drawn as ovals
- Each attribute is linked to its entity by a line
- The name of the attribute is written in the oval



Diagramming Relationships

- Relationships are links between two entities
- The name is given in a diamond box
- The ends of the link show cardinality



3.2. ER-to-Relational Mapping Algorithm

Case Study (Requirement Analysis for a company)

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 stated the following description of the "miniworld"—the part of the company to be represented in the database:

1. 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.
2. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
3. We store each employee's name, social security number, address, salary, sex, 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 keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.

4. We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent’s first name, sex, birth date, and relationship to the employee.

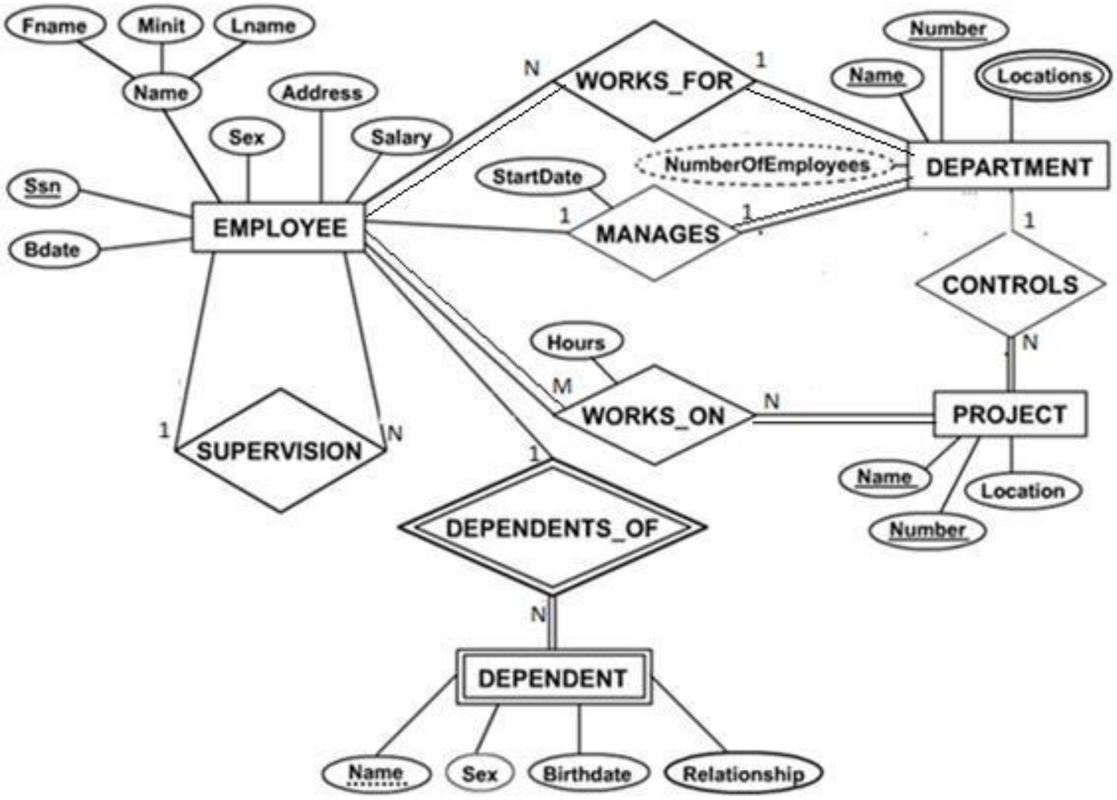


Figure 3.1: ER-diagram for a company

Step 1: Mapping of Strong 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. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as 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: For the company database ER

- ✓ Regular entities are:

EMPLOYEE, DEPARTMENT, and PROJ ECT

- ✓ The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps.

- ✓ In our example, we choose SSN, DNUMBER (Department Number), and PNUMBER (Project Number) as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT,

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 and include all simple attributes (or simple components of composite attributes) of W as attributes of R. In addition, include as foreign key attributes of R the primary key attributes) of the relations) that correspond to the owner entity types): this takes care of the identifying relationship type of W The primary key of R is the combination of the primary keys) of the owners) and the partial key of the weak entity type W, if any.

- ✓ In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.
- ✓ We include the primary key SSN of the EMPLOYEE relation which corresponds to the owner entity type-as a foreign key attribute of DEPENDENT;
- ✓ The primary key of the DEPENDENT relation is the combination {SSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

Step 3: Mapping of Binary 1:1 Relationship Types

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

(1) The foreign key approach,

(2) The merged relationship approach, and

(3) The cross-reference or relationship relation approach.

Approach 1 is the most useful and should be followed unless special conditions exist, as we discuss below.

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

- ✓ In our example, we map the 1:1 relationship type MANAGES by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it MGRSSN (manager social security number). We also include the simple attribute STARTDATE of the MANAGES relationship type in the DEPARTMENT relation and rename it MGRSTARTDATE.

Note that it is possible to include the primary key of S as a foreign key in T instead. In our example, this amount to having a foreign key attribute, say DEPARTMENT_MANAGED in the EMPLOYEE relation, but it will have a null value for employee tuples who do not manage a department.

2. Merged relation option: An alternative mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.

3. Cross-reference or relationship relation option: The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types. As we shall see, this approach is required for binary M: N relationships. The relation R is called a relationship relation, (or sometimes a lookup table), because each tuple in R represents a relationship instance that relates one tuple from S with one tuple of T.

Step 4: Mapping of Binary 1: N Relationship Types

For each regular binary 1: N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; this is done because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1: N relationship type as attributes of S.

In our example, we now map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION .

For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO (since we cannot give them the same name in the relation).

For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself because the relationship is recursive-and call it SUPERSSN.

The CONTROLS relationship is mapped to the foreign key attribute DNUM of PROJECT, which references the primary key DNUMBER of the DEPARTMENT relation.

Step 5: Mapping of Binary M: N Relationship Types

For each 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. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate relationship relation S.

In our example, we map the M: N relationship type WORKS_ON by creating the relation WORKS_ON.

We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them PNO and ESSN, respectively.

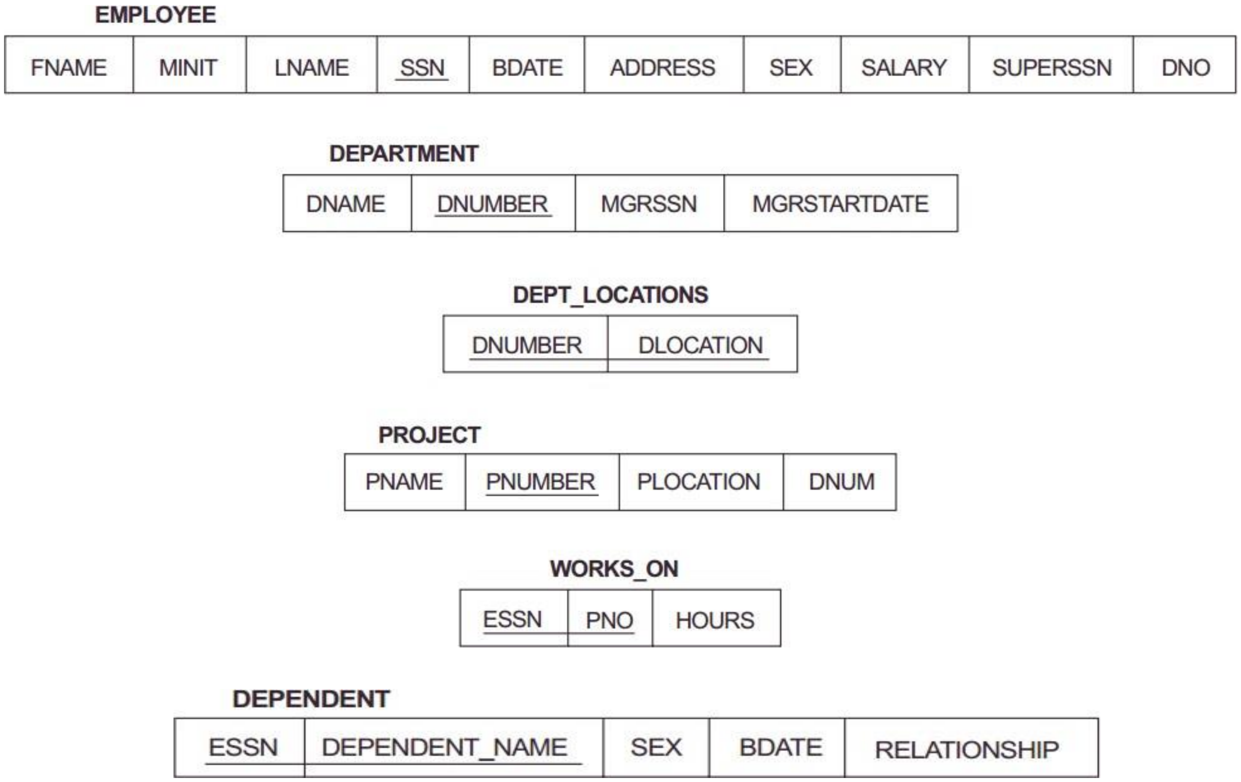
We also include an attribute HOURS in WORKS_ON to represent the HOURS attribute of the relationship type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

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 or 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.

In our example, we create a relation DEPT_LOCATIONS. The attribute DLOCATION (Department Location) represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key represents the primary key of the DEPARTMENT relation.
The primary key of DEPT_LOCATIONS is the combination of {DNUMBER, DLOCATION}. A separate tuple will exist in DEPT_LOCATIONS for each location that a department has.

Therefore, finally we will end up with the following database schema of company.



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

3.3. Problem with E-R models

The ER model is incomplete as any other data model. It is necessary to understand the reason for its incompleteness because only then can we develop approaches to overcome the model's limitations. Sometimes, a problem known as a connection trap occurs due to the misinterpretation of the meaning of certain relationships.

There are two main types of connection models are:

1. **Fan Traps**: It occurs when a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous.
2. **Chasm Traps**: It occurs when a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences.

Both Fan and Chasm traps occur when it becomes impossible to retrieve all the necessary information needed from the entire diagram.