Chapter 4

# DATA MODELS

## 4.1 INTRODUCTION

A data model provides mechanism to structure the data for the entities being modelled, allow a set of manipulative operators to be defined on them, and enforce set of constraints to ensure the accuracy of data. Thus we can say it is a conceptual method of structuring the data.

The data models can be categorized into three major categories according to the types of concepts they use to describe the database structure. These are:

(*a*) **High-level or conceptual data models :** These provide the concepts that are close to the way many users perceive data. These data models use concepts such as entities, attributes and relationships.

(*b*) **Low level or physical data models :** These provide concepts that describe the details of data stored in the computer by representing information such as record formats, record orderings, and access paths. The concepts provided by this model are usually meant for specialized users (DBA etc.) not for end users.

(*c*) **Representational or record based data models :** These represent data by using record structures and are generally used in commercial DBMS's. These data models hide some details of data storage and can be implemented on a computer system directly. The concepts provided by these models may be understood by end users. These are further categorized into **network model, hierarchical model** and **relational model.**

In this chapter, we discuss, the various representational or record based data models in detail with their advantages and disadvantages. The relational model is discussed in more detail due to its popularity and ease of use.

## 4.2 HIERARCHICAL MODEL

Hierarchical model is based on tree structure. A hierarchical database consists of collection of records, that are connected to each other by links.

**Record** : A record is a collection of attributes, each contains only one data value.

**Link** : A link is an association between two records.

The tree structure used in hierarchical model is known as rooted tree. The Root node of that tree is dummy node or an empty node. So, hierarchical model is a collection of **rooted trees**. Collection of rooted trees make forest. A rooted tree is also known as **database tree**.

### 4.2.1 Tree Structure Diagrams

Tree structure consists of two basic components:

1. **Rectangular boxes** : Rectangular boxes represent various record types.
2. **Line** : Line represents link between two record types.

A tree structure diagram specifies the overall logical structure of database (as E-R diagram in Entity Relationship model).

Relationship exists between a parent and a child must be one-to-many or one-to-one relationship.

Link between parent and child is represented by line with an arrow. A parent may have an arrow pointing to child, but child must have an arrow pointing to its parent.

A general tree structure is shown in Figure 4.1.

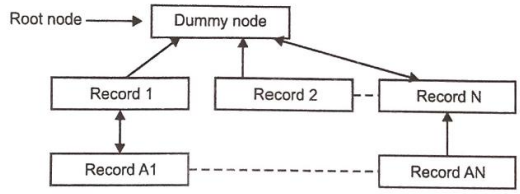


**FIGURE 4.1.** General tree structure.

**Example.** Consider the relation **working-for** between **Employee** and **Department**

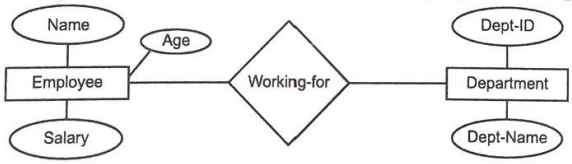


**FIGURE 4.2.** E-R diagram of working-for.

Record Employee consists of three attributes (Name, Age and Salary) and record Department consists of two attributes (Dept-ID and Dept-Name). An E-R diagram for this relation is shown in Figure 4.2.

(i) First, suppose an employee can work in only one department but any department can have more than one employee.

Corresponding tree structure diagram is shown in Figure 4.3(*a*) and sample database in Figure 4.3(*b*).

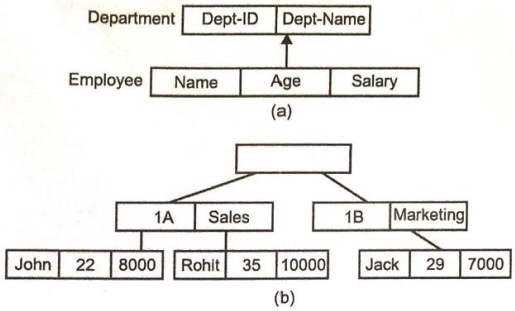


**FIGURE 4.3**

(ii) Now suppose an employee can work only in one department as well as every department can have only one employee.

Corresponding tree structure diagram is shown in Figure 4.4(*a*) and sample database in Figure 4.4(*b*).

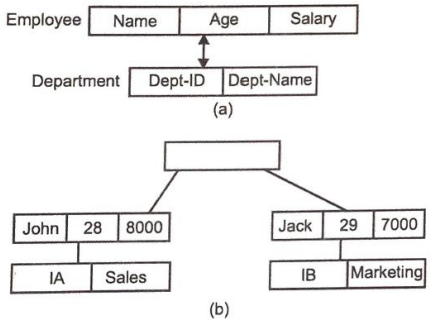


**FIGURE 4.4**

(iii) Now suppose an employee can work in more than one department and any department can have more than one employee. In that case we have to make two separate tree structure diagrams *because Many-to-Many relationship is not allowed in tree structure diagram.*

Corresponding tree structure diagrams are shown in Figure 4.5(*a*), (*b*).

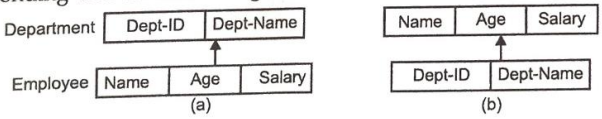


**FIGURE 4.5.** Tree structure diagram.

Corresponding sample database for tree structure diagrams in Figure 4.5(*a*), (*b*) are shown in Figure 4.6(*a*), (*b*) respectively.

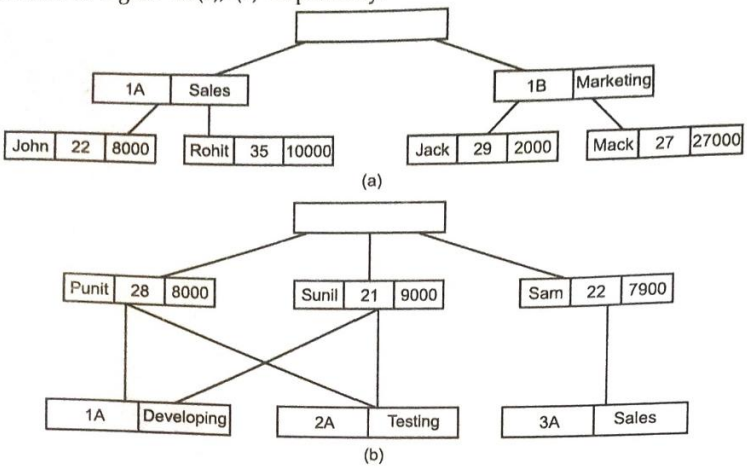


**FIGURE 4.6.** Sample database.

In this case, the sample database consists of more than one tree structure diagram.

(*iv*) In case of descriptive attributes, tree structure diagram is more complicated. Consider the E-R diagram shown in Figure 4.7.

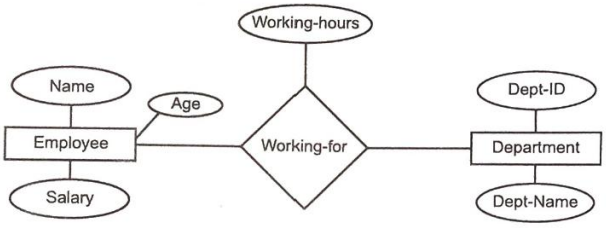


**FIGURE 4.7.** E-R diagram.

Because a **link** cannot have any value, so we have to make new record for descriptive attribute. Corresponding tree structure diagrams for E-R diagram in Figure 4.7 are shown in Figure 4.8(*a*), (*b*). Assume conditions of 4.2.1 (*iii*).

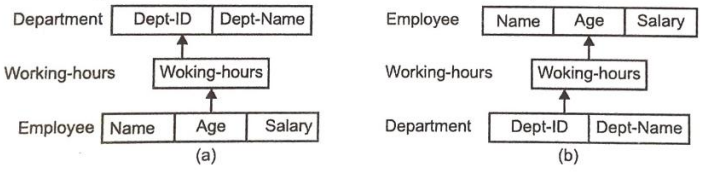


**FIGURE 4.8.** Tree structure diagram.

Corresponding sample database for tree structure diagrams in Figure 4.8(*a*), (*b*) are shown in Figure 4.9(*a*), (*b*) respectively.

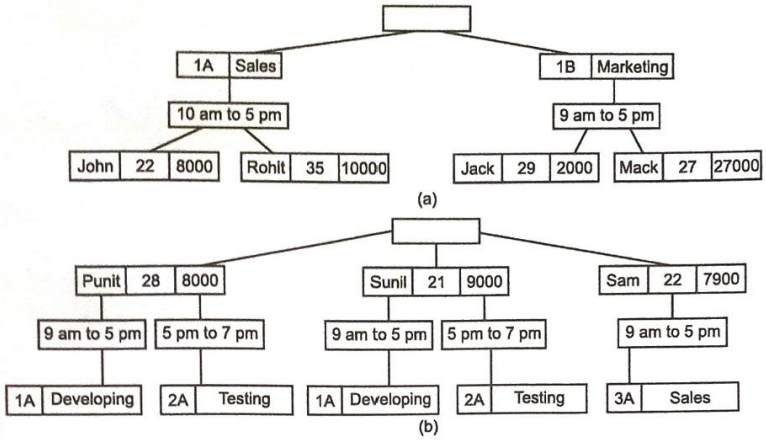


**FIGURE 4.9.** Sample database.

### 4.2.2 Operations on Hierarchical Data Model

The basic operations that can be performed on Hierarchical data model are insertion, deletion, updation and retrieval. All these operations discussed briefly.

1. **Insertion Operation** : The Insert Operation is used to insert a new record into the database. The newly inserted record becomes the current record for the database.

   If the Inserted Record is a root record then it creates new hierarchical tree with the new record as the root. But if it is a child record then we should make its parent first because a child node cannot exist without a parent (root).

2. **Deletion Operation** : The delete operation is used to delete a record from the database. To delete a record from the database, we first make it the current record and then issue the delete command.

3. **Updation Operation** : The updation operation is used to update a record in the database.

4. **Retrieval Operation** : The process of searching and fetching of a record in the database is known as retrieval of a record.

### 4.2.3 Query Language for Hierarchical Databases

Consider the example of **Employee—Department** relation schema.

**Program Work Area**

Program work area is a buffer storage area which contains the following variables.

(*i*) *Record template* : Record template is a record for each record type for example Employee record for Employee record type, Department record for department record type.

(ii) *Currency pointers* : It is a set of pointers, one for each database tree which contains the address of most recently used record.

(iii) *Status flag* : It is a variable which represents the result of most recent database operation.

**Get Command**

Get Command is used for data retrieval. The general format of Get command is

```
Get <record type>
    Where <condition>
```

There are two types of Get command. These are:

(i) *Get first* : This command gives the first record of a given record type from the database which satisfies the given condition. The general format of **Get first** command is

```
Get first <record type>
          Where <condition>
```

*Ex.*

```
Get first <Employee>
          Where employee. Dept–name = "Developing";
            gives employee Punit.
```

(ii) *Get next* : This command gives the next record of a given record type from the database which satisfies the given condition. The general format of Get next command is

```
Get next <record type>
         Where <condition>
```

*Ex.*

```
Get next <Employee>
         Where employee. Dept-name = "Developing";
           gives employee Sunil.
```

If you want to locate a record within the same parent then use the command

```
Get next within parent <record type>
                       Where <condition>
```

**Update Commands**

(i) *Insert command* : This command is used to insert a new record of a given record type into the database. The general format of insert command is

```
insert <record type>
       Where <condition>
```

*Ex.* For adding a new employee we write the query

```
employee.name = "Rahul";
employee.age = "25";
employee.salary = "8000";
Insert employee
    Where department.Dept-Id = "3A";
```

(ii) *Replace command* : This command is used to modify an existing record in database. The general format of **replace** command is

```
get hold first <record type>
      where <condition>
-----------------------------------
replace;
```

*Ex.*

```
get hold first <employee>
                where employee.Name = "Sam";
                      employee.salary = "9000";
replace;
```

(iii) *Delete command* : This command is used to delete an existing record in database. The general format of delete command is

```
get hold first < record type>
                where <condition>;
delete;
```

*Ex.*

```
get hold first <employee>
                where employee.Name = "Sunil";
delete;
```

Delete command not only deletes a parent but also deletes all of its children.

### 4.2.4 Virtual Records

In tree structure diagram, you cannot represent many-to-many relationship directly. To represent these relations and keep tree-structure organization you have to replicate data. To overcome the drawbacks of data replication virtual records are used.

A Virtual record is a record with no data values but it contains a pointer to the physical record. To avoid replication, keep a single record and place virtual record instead of actual record. Corresponding tree structure diagram with virtual records of Figure 4.9(*b*) is shown in Figure 4.10.

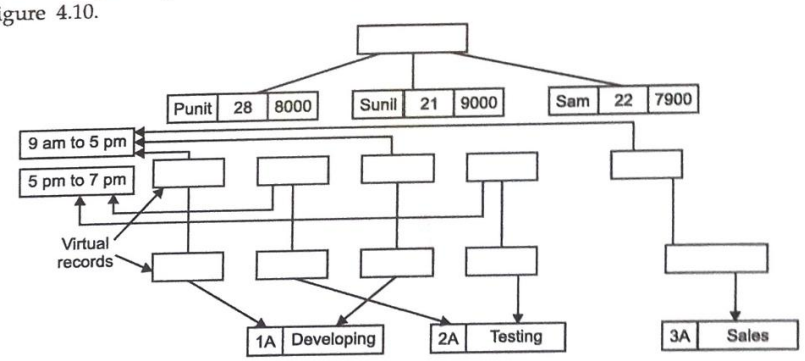


**FIGURE 4.10.** Tree structure diagram with virtual records.

Virtual records keep data consistency but wastage of storage space is still a serious problem.

### 4.2.5 Implementation of Tree Structure Diagram

To optimize tree structure diagram, **leftmost–child, preorder threads** and **next-sibling pointers** are used instead of parent-child pointers. Consider Figure 4.6(*a*), the corresponding optimized tree structure diagram is shown in Figure 4.11.

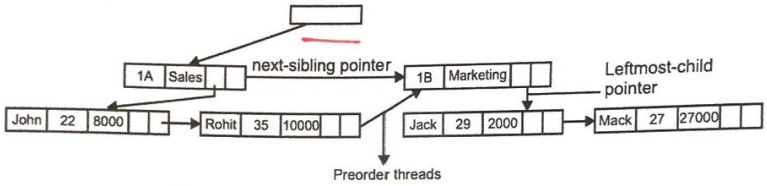


**FIGURE 4.11.** Optimized tree structure diagram.

#### Advantages of Hierarchical Model

The following are the main advantages of hierarchical data model:

1. **Simplicity** : In this model, records are related in form of parent/child relationship. So performing various operations (*i.e.*, insertion, deletion etc.) in this tree like structure is easy and simple to perform. This results in the simple design of the database resulting from this model.
2. **Integrity of Data** : The parent/child relationship between the various records in the hierarchical model is represented by a relationship or link. Each child segment can be linked to only one parent and a child can only be reached through it parent, so this model promotes data integrity.
3. **Data Security** : Each child segment can be linked to only one parent and a child can only be reached through its parent in this model. So for deleting the child segment proper information of parent segment is needed. Thus it provides data security which is enforced by the DBMS.
4. **Efficiency** : The hierarchical model contains one to many relationships between parent and child. When the database contains many 1 : N relationships between various records then this model handles it very efficiently.
5. It is very efficient to handle large number of transactions using this model. This is mainly because the links (or relationship) established by the pointer in the various records are permanent and cannot be modified.

#### Disadvantages of Hierarchical Model

The information is replicated in hierarchical database. The replication may occur either in different database trees or in same tree. Consider Figure 4.9(*b*), where records (1A, developing) and (2A, testing) are replicated. The other disadvantages are as follows:

1. **Knowledge of physical level of data storage is required** : The requirement for a one to many relationship between parent and child can result in redundancy of data.

To get around the redundancy problems, data is stored in one place and referenced by links or physical pointers, which requires technical skills.

2. **Complexity** : The physical links make it very difficult to expand or modify the database, changes typically require substantial rewriting efforts.
3. **Inflexibility** : The basic problem occurs with this model is that they are not flexible enough to establish all the relationships (many-to-many etc.) which occur in the real world. Usually there are one to many relationship between the records, established by pointers which are permanent and cannot be modified in case of other cases where relationships (like many to many etc.) exist.
4. **Lack of querying facilities** : The lack of declarative querying facilities and need for navigation of pointers to access needed information make querying rather complex. It does not provide the adhoc query capability easily.
5. **Database management problems** : In this model, the modifications to the data structure leads to significant modifications to application programs that access the database. Also new relations or nodes result in complex system management tasks.
6. **Problems with data manipulation operations** : Various problems are encountered while performing various operations like insertion, deletion and updation. Moreover the data retrieval is also very complex and asymmetric. Therefore, a better model is needed to solve these problems.
7. **Lack of standards** : This model does not have any specific or precise standard for database design and modelling.

NOTE *Record-type is equivalent to name of table in relational model and entity set in E-R model.*
*Record is equivalent to tuple in relational model and entity in E-R model.*
*But generally we use term record instead of record-type.*
*This is applicable for both hierarchical model and network model.*

### Commercially Available Hierarchical Database Systems

There are number of Database systems based on hierarchical model. Some of them are as follows :

1. IBM's Information Management System.
2. MRI's System 2000
3. IMS Informatics Mark IV.
4. Time-shared Data Management System (TDMS) of SDC.

## 4.3 NETWORK MODEL

Network model is based on graph structure. A network database consists of collection of records, which are connected to each other by links.

**Record** : A record is a collection of attributes, each contain only one data value.

**Link** : A link is an association between two records.

So, network model is a collection of graphs.

### 4.3.1 Graph Structure Diagrams

Graph structure consists of two basic components.

(*i*) **Rectangular boxes** : Rectangular boxes represent various record types.

(*ii*) **Line** : Line represents link between two records. A link cannot contain data value.

A graph structure diagram specifies the overall logical structure of database.

**Example.** Consider the relation **Working-for** between **Employee** and **Department.**

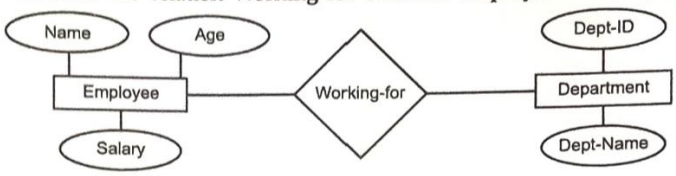


**FIGURE 4.12**

(*i*) First suppose an employee can work only in one department but any department can have more than one employee.

Corresponding graph structure diagram is shown in Figure 4.13(*a*) and sample database in Figure 4.13(*b*).

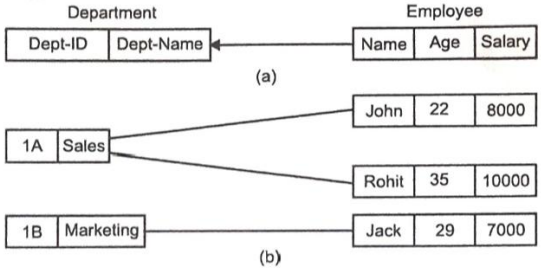


**FIGURE 4.13**

(*ii*) Now suppose an employee can work only in one department as well as every department can have only one employee.

Corresponding graph structure diagram is shown in Figure 4.14(*a*) and sample database in Figure 4.14(*b*).

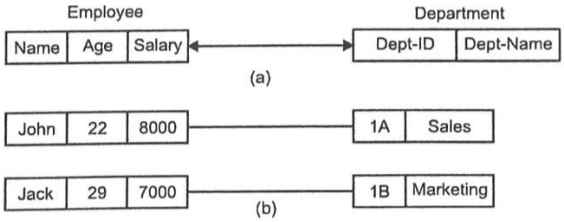


**FIGURE 4.14**

(*iii*) Now suppose an employee can work in more than one department and any department can have more than one employee.