Department of Distance Education

Punjabi University, Patiala

Class : B.A. II (Computer Applications) Semester : 4

Paper : (BAP 203) Data Base Management System (DBMS)

Medium : English Unit : I

Lesson No.

INTRODUCTION

Structure:

1.1.0 Introduction:

Data is termed as the collection of meaningful unorganized facts, concepts or instructions. Information is processed form of data. Traditional File Processing System is a method for storing and organizing data stored in computer files. But Traditional File Processing has significant disadvantages of data redundancy, lack of flexibility, data dependency etc.

In this lesson, we first provide the formal definitions of data, information, and traditional file processing system. Then we define the limitations of traditional file processing system and finally discuss the concepts of database

1.1.1 Objectives

After completing this lesson, you will be able to:

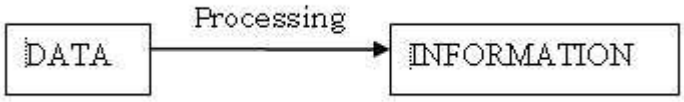
- Define data and information
- Discuss traditional file processing system, its characteristics and limitations
- Define database and explain its components

1.1.2 Data and Information

Data is defined as a collection of meaningful facts which can be stored and processed by computer or humans. The main examples of data are names, phone numbers, marks, age of employees etc.

Information is processed form of data which helps us in making decisions. For examples: the roll no and marks of all the students is the data and when it processed and report cards of the students are prepared for taking the decision that which student stood first, second and third in the class , then it becomes the information.

Pictorially we can show it as:



The term **data** refers to factual **information**, especially that used for analysis and based on reasoning or calculation. **Data** itself has no meaning, but becomes **information** when it is interpreted. **Information** is a collection of facts or **data** that is communicated. However, in many contexts they are considered and are used as synonyms. **Data**, by the way, is the plural of *datum*. **Information** comes from Latin *informationem* 'concept, idea' or 'outline'.

The relationship between data and information works this way:

Data = the facts about a topic.

Information = evaluated data used to answer a question.

1.1.3 Traditional File Processing System:

Let's take the example of our university to explain this system. Our objective is to computerize the records of various departments of the university. By not going so deep into the records, we just try to computerize the records of employees of the university. In the past system, records were stored on the registers manually. Now in this system, we are using computer for storing the records of employees. For storing the records, we use files in computer. Each file has a specified format for storing the file. Suppose we are only storing the name, date of birth, date of joining, department, permanent address, correspondence address of employee in the file. So, all these are fields of a record. The information of all fields for a particular employee will become a record of a particular employee in that file. All the fields are separated by a tab in a file and one record is stored in a line in file.

Thus, for example following will be the format of the file containing the records of the university employees maintained by the establishment department of the university:

S.No.	Name	DOB	DOJ	Department	Permanent Address	Correspondence address
1.	Vishal	23-11-1977	23-02-2005	Computer Sc.	Peeran wala mohalla, ferozepur city	Lecturer (CS), Dept of Computer Sc., P.U.Patiala

.....
.....
.....

In the similar manner, Department of Computer Science (DCS) will also maintain the record of its employee for recording the day to day activities of the employees. Thus this file maintained by DCS will also contain the name, DOB, DOJ, permanent address, correspondence address and other field specific to department.

Formally, **a record** is a collection of related fields that are treated as a single unit. Further all the records are grouped together to form **a file**. All the related files are grouped together to form **a database**. Thus, in the above example, all the files of the university are interrelated to each other, when grouped together they form database of the university. The database of the university consists of account file, student record file, employee record file etc.

So, **Traditional file processing system** has for each application a separate master file and its own set of personal files. Such file based approaches which came into being with the first commercial application of computers did provide an increased efficiency in the data processing compared to earlier manual paper record-based systems. As the demand for efficiency and speed increased, the computer-based simple file oriented approach to information processing started suffering from number of limitations which is explained in the following section.

1.1.4 Limitations of Traditional File Processing System

Following are the limitations of traditional file processing system:

1. **Data Redundancy:** By 'Data Redundancy' we mean the duplication of same data at various places. In Traditional File Based Systems, each application has its own private files. So, same data is stored in different files. For example, In the above example of university, Department is storing the personal details of employees and Accounts Department is also storing the personal details of the employees for their purpose. This fact leads to considerable redundancy in stored data, with resultant wastage in storage space.
2. **Data Inconsistency:** By 'Data Inconsistency' we mean that same data stored in different files do not match with each other at particular moment of time. For example, when the employee had joined the university, he has given some correspondence address. But after some period of time, his correspondence address changes and he has informed his department only and forgets to inform other departments. Thus one file (maintained by his own department) is updated with new correspondence address, but the other files (maintained by other departments) are still storing the obsolete correspondence address of that employee. This is known as Data Inconsistency.
3. **Difficulty in sharing data:** In this system, each file has its own format for storing data. By Format we mean that delimiters separating each record in a file and further for each field in a record can or cannot be same for different files in an application. Thus for using the data from other file in a file requires to know the format of the file from which the data is to be shared. So, it becomes very difficult.
4. **Incompatible File Formats:** Each programmer stores the data in the files in the format as per his choice as there is no standard file format for storing the files. Thus, it becomes very difficult to share data among the files having different format. Even If some other programmer has to work on a file stored by different programmer, he hasto first understand the file format of that file and then start working on that file.

5. Lack of Data Security: The data stored in the database must be protected from unauthorized access. Since in Traditional File Processing System, the application programs are added to the system on the basis of queries which are not predefined so it is difficult to enforce security measures on these application programs.

6. Lack of data integrity: Data Integrity means data correctness. For example: Basic Pay cannot be negative. Such type of data constraints can be imposed in the traditional file processing system by writing appropriate code in the application programs. But in future, if more constraints are to be added then, we need to modify the application program and sometimes it is not possible to change the application code. Thus it may lead to poor data which may result in bad/wrong decisions.

7. Data Dependency: Data Dependence means it is impossible to change the storage structure without affecting the application program. For example, if we change the delimiter separating the fields in the records of a file from tab to double space, we must change the code in the application program that access data from that file as the structure of file gets changed. Due to this we cannot change the storage structure without changing the application code.

8. Lack of Flexibility: In Traditional File Processing System, programmers are already told about which type of queries need to be answered by the application. And programmers code that query using the data files for that application. But in the fast moving and competent business environment of today, apart from such regular queries, there is a need for responding to un-anticipatory queries, and then the system will fail. Then the programmer again has to code for such queries. Thus this limitation leads to lack of flexibility of the file system.

9. Inadequate data modeling of real world: The file system approach has inability to design a database which shows the basic entities, relationships and events facing the organization every day. Complex data and interfile relationships cannot be formally defined to the system.

10. Concurrency Problem: Concurrency means simultaneous access of the same file by two or more users. When data in a file is simultaneously accessed by two or more users for updating the data, there must be a system that finally does not lead to inconsistent data. But in this file system, it is not possible to implement such feature. If possible, it is very difficult to implement.

1.1.5 Database and Database System: Database is collection of related data. The database can be of varying sizes and varying complexity. In order to overcome the limitations of the traditional file processing system, the concept of Database Systems was introduced. A database system is basically a computerized record keeping system i.e. it is a computerized system whose overall purpose is to maintain information and to make the information available on demand.

In other words, a database is a collection of interrelated data stored in a

database server. These data will be stored in the form of tables. The primary aim of database is to provide a way to store and retrieve database information in fast and efficient manner. There are number of characteristics that differ from traditional file management system. In file system approach, each user defines and implements the needed files for a specific application to run. For example in sales department of an enterprise, one user will be maintaining the details of how many sales personnel are there in the sales department and their grades. These details will be stored and maintained in a separate file. Another, (d) user will be maintaining the salesperson salary details working in the concern. The detailed salary report will be stored and maintained in a separate file. Although both the users are interested in the data of the salespersons they will be having their details in a separate file and they need different programs to manipulate their files. This will lead to wastage of space and redundancy or replication of data, which may lead to confusion. Sharing of data among various users is not possible due to which, data inconsistency may occur. These files will not be having any inter-relationship among the data stored in these files. Therefore in traditional file processing, every user will be defining his own constraints and implement the files needed for the applications.

In database approach, a single repository of data is maintained that is defined once and then accessed by many users. The fundamental characteristic of database approach is that the database system not only contains data but it contains complete definition or description of the database structure and constraints. These definitions are stored in a system catalog, which contains the information about the structure and definitions of the database. The information stored in the catalog is called the metadata, it describes the primary database. Hence this approach will work on any type of database for example, insurance database, Airlines, banking database, Finance details, and Enterprise information database. But in traditional file processing system the application is developed for a specific purpose and they will access specific database only.

The other main characteristic of the database is that it will allow multiple users to access the database at the same time and sharing of data is possible. The database must include concurrency control software to ensure that several users trying to update the same data at the same time, do so in a controlled manner. In file system approach many programmers will be creating files over a long period and various files have different format in various application languages.

Therefore there is possibility of information getting duplicated. This redundancy in storing same data multiple times leads to higher costs and wastage of space. This may result in data inconsistency in the application; this is because update is done to some of the files only and not all the files. Moreover in database approach multiple views can be created. View is a tailored representation of information contained in one or more tables. View is also called as “Virtual table”

because view does not contain physically stored records and will not occupy any space.

A multi-user database whose users have variety of applications must provide facilities for defining multiple views. In traditional file system, if any changes are made to the structure of the files, it will affect all the programs, so changes to the structure of a file may require changing all the programs that access the file. But in case of database approach the structure of the database is stored separately in the system catalog from the access of the application programs. This property is known as program-data independence.

Database can be used to provide persistent storage for program objects and data structures that resulted in object oriented database approach. Traditional systems suffered from impedance mismatch problem and difficulty in accessing the data, which is avoided in object oriented database system. Database can be used to represent complex relationships among data as well as to retrieve and update related data easily and efficiently.

It is possible to define and enforce integrity constraints for the data stored in the database. The database also provides facilities for recovering from hardware and software failures. The backup and recovery subsystem is responsible for recovery. It reduces the application development time considerably when compared to the file system approach and there is availability of up-to-date information of all the users. It also provides security to the data stored in the database system.

Example: University Database

Used to maintain information concerning students, courses and grades in a university environment.

The database has five files, each of which stores data records of the same type.

The **Student** file stores data of each student,

The **Course** file stores data on each course,

The **Section** file stores data on each section of a course,

The **Grade_Report** stores information on the grades students receive in the various sections they have completed,

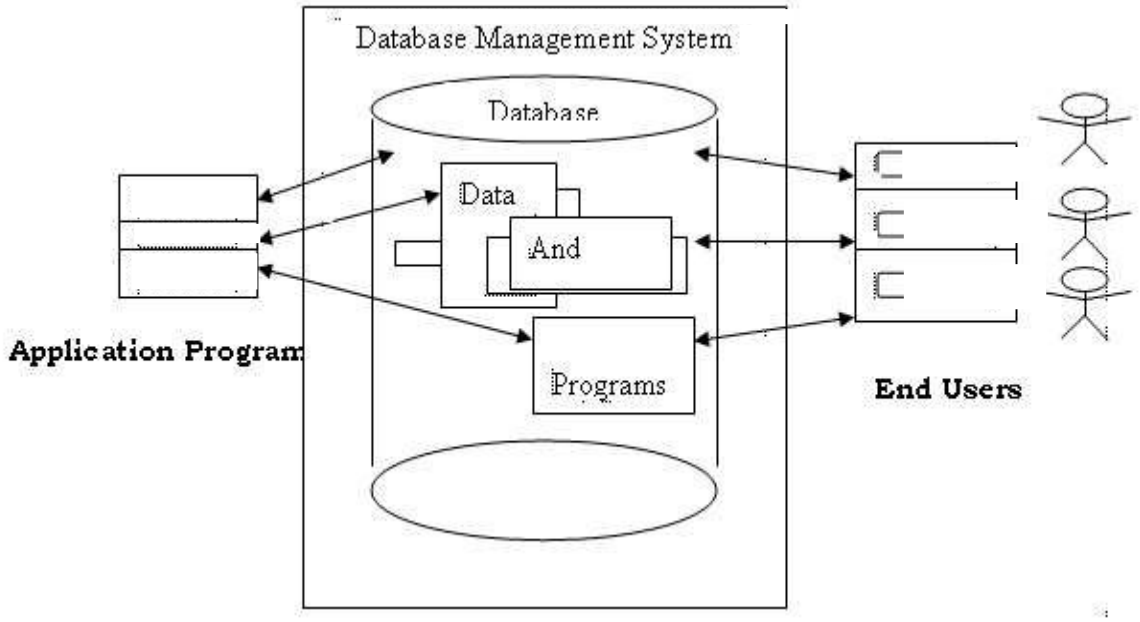
The **Prerequisite** file stores the prerequisite of each course.

Database Approach vs. Traditional File Processing

- Self contained nature of database systems (database contains both data and meta-data).
- Data Independence: application programs and queries are independent of how data is actually stored.
- Data sharing.
- Controlling redundancies and inconsistencies.
- Secure access to database; Restricting unauthorized access.
- Enforcing Integrity Constraints.

- Backup and Recovery from system crashes.
- Support for multiple-users and concurrent access.

1.1.6 Components of Database System: A database system comprises of four major components namely, data, hardware, software, and users. The simplified view of a database system is as follows:



Application Programs
End Users

Data: It is very important component of the database system. Most of the organizations generate, store and process large amount of data. The users which directly access it or access it through some application programs. **In general, the data in the database will be both integrated and shared.** Let us explain why we mean by the terms "integrated and shared":

By Integrated we mean that the database can be thought of as unification of several distinct files, with any redundancy among those files wholly or partly eliminated.

By shared, we mean that individual pieces of data in the database can be shared among several different users, in the sense that each of those users can have access to the same piece of data. Such sharing is the consequence of the fact that database is integrated.

Hardware:

The hardware of the database system consists of:

1. The secondary storage devices, usually, magnetic disks, hard disks,

CD-ROMs, Floppy Disks - that are used to hold the stored data, together with the associated I/O devices, device controllers etc.

2. The processor(s) and associated main memory that are used to support the execution of the database system software.

Software:

Software layer is present in the database system between the physical database itself i.e. where the data is actually stored and the users of the system and is known as Database Management System (DBMS). All the requests from the user for access to the database are handled by DBMS. One general function of DBMS is thus the shielding of database users from hardware level details. Basically, DBMS acts like a bridge between users and database. Database management systems are programs that are written to store, update, and retrieve information from a database. There are many databases available in the market. The most popular are the Oracle and SQL Server. The Oracle database is from the Oracle Corporation and the SQL Server is from the Microsoft Corporation. There are freely available database like MySQL. These are open source databases. Database Management Systems are available for personal computers and for huge systems like mainframes. DB2 is a database from IBM for Mainframe systems.

Users:

There are a number of users who can access or retrieve data on demand using application programs and interfaces provided by DBMS. Each type of user needs different software capabilities. Following are the categories of the users:

1. Application Programmers
2. End Users
3. Database Administrator (DBA)

Application Programmers: Users who are responsible for writing application programs that use database. These programs operate on data for retrieving existing information, inserting new information, deleting or changing existing information.

End Users: End Users are those who need not know about the presence of database system or any other system supporting their usage. These users interact with the system via the interface (menu- or- form driven) provided by DBMS. For example: Users of Automatic Teller machines (ATM) falls under this category.

Database Administrators (DBA): The database administrator is the person who implements the strategic and policy decisions regarding the data of the enterprise. Thus DBA is responsible for the overall control of the system at a technical level. Detailed responsibilities of DBA will be discussed in Lesson 3.

1.1.7 Summary:

Data is defined as a collection of meaningful facts which can be stored and processed by computer or humans. **Information** is processed form of data which helps us in making decisions. **Traditional file processing system** has for each

application a separate master file and its own set of personal files. **A record** is a collection of related fields that are treated as a single unit. Further all the records are grouped together to form **a file**. All the related files are grouped together to form **a database**. The computer-based simple file oriented approach to information processing started suffering from number of limitations like data redundancy, data inconsistency, difficulty in sharing data, concurrency problems etc. In order to overcome the limitations of the traditional file processing system, the concept of Database Systems was introduced. A database system is basically a computerized record keeping system i.e. it is a computerized system whose overall purpose is to maintain information and to make the information available on demand. A database system comprises of four major components namely, data, hardware, software, and users.

1.1.8 Self Understanding:

Q1. Explain the limitations of the Traditional file Processing System?
Q2. What is a database and database System?
Q3. Explain the Components of a database system?
Q4. Discuss the categories of users in a database system.
Q5. Define DBMS.
Q6. Define file, record and field.

1.1.9 Further Readings:

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

DATABASE MANAGEMENT SYSTEM-I

Structure:

1.2.0 Introduction:

In this lesson, we first provide the formal definitions of Database Management System. Then we define the Characteristics of DBMS and finally we will explain the advantages of DBMS over traditional file processing system.

1.2.1 Objectives

After completing this lesson, you will be able to:

- Define DBMS and its Characteristics
- Advantages of DBMS over Traditional File Processing System

1.2.2 The Database Management System (DBMS):

A Database Management System is the software system that allows users to define, create and maintain a database and provides controlled access to the data.

It is basically a collection of programs that enables users to store, modify and extract information from a database as per the requirements. DBMS is an intermediate layer between programs and the data. Programs range from small systems that run on personal computers to huge systems that run on mainframes. Some of common examples of database applications are as follows:

1. Billing System at Super Stores
2. Patient Management System
3. Student Record Management System
4. Computerized Account Department at Educational Institutes
5. Computerized Flight Reservation System

The DBMS relieves the user from knowing how data is stored physically and the complex algorithms used for performing operations on the database.

Conceptually, what happens is the following:

1. A user issues an access request, using some particular data sublanguage (DDL, DML, and DCL using SQL).

2. The DBMS intercepts that request and analyzes it.
3. The DBMS inspects, in turn, the external schema for that user, the corresponding external/conceptual mapping, the conceptual schema, the conceptual/internal mapping, and the storage structure definition.
4. The DBMS executes the necessary operations on the stored database.

In other words, DBMS is a collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMSs, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

- computerized library systems
- automated teller machines
- flight reservation systems
- computerized parts inventory systems

From a technical standpoint, DBMSs can differ widely. The terms relational, network, flat, and hierarchical all refer to the way a DBMS organizes information internally. The internal organization can affect how quickly and flexibly you can extract information.

Requests for information from a database are made in the form of a query, which is a stylized question. For example, the query

SELECT ALL WHERE NAME = "SMITH" AND AGE > 35

requests all records in which the NAME field is SMITH and the AGE field is greater than 35. The set of rules for constructing queries is known as a query language. Different DBMSs support different query languages, although there is a semi-standardized query language called SQL (structured query language). Sophisticated languages for managing database systems are called fourth-generation languages, or 4GLs for short.

The information from a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report. Many DBMSs also include a graphics component that enables you to output information in the form of graphs and charts.

1.2.3 Characteristics of DBMS:

Following are the characteristics that a DBMS must include:

1. DBMS must be able to accept data definitions (external schemas, the conceptual schemas, and all associated mappings) in source form and convert them to the appropriate object form. In other words, the DBMS include language processor components for each of the various data definitions languages (DDLs). The DBMS must also "understand" the DDEL definitions, in the sense that, for example, it "understands" the EMPLOYEE external records include a SALARY field; it must then be able to use this knowledge in interpreting and responding to user requests.
2. The DBMS must be able to handle requests from the user to retrieve, update, or delete existing data in the database, or to add new data to the database. In

other words, the DBMS must include a data manipulation language (DML) processor component.

3. The DBMS must monitor user requests and reject any attempt to violate the security and integrity rules defined by the DBA.
4. The DBMS must enforce certain recovery and concurrency controls.
5. The DBMS must provide a data dictionary function. The data dictionary can be regarded as a database in its own right. The dictionary contains "data about the data" i.e. definitions of other objects in the system – rather than just "raw data". In particular, all the various schemas and mappings will physically be stored, in both source and object form, in the dictionary. The dictionary might even be integrated into the database it defines, and thus include its own definition. It should certainly be possible to query the dictionary just like any other database, so that, for example, it is possible to tell which program and/ or user are likely to be affected by some proposed change to the system.
6. DBMS should perform all of the functions as efficiently as possible.
7. Data should be stored with minimum redundancy to ensure consistency in the data across different applications.

1.2.4 Advantages of DBMS over Traditional File Processing System:

1. Redundancy can be reduced: In Traditional File Processing System, each application has its own private files, which cannot be shared between multiple applications. This can often lead to considerable redundancy in the stored data, which results in wastage of storage space. By having centralized database most of this can be avoided. It is not possible to eliminate all the redundancy. Sometimes there are sound business and technical reasons for maintaining multiple copies of the same data. In a database system, however this redundancy can be controlled.

For example: In case of college database, there is some common data of the student which has to be mentioned in each application, like Rollno, Name, Class, Phone no, Address etc. This will cause the problem of redundancy which results in wastage of storage space and is difficult to maintain, but in case of centralized database, data can be shared by number of applications and the whole college can maintain its computerized data with the database containing Roll no, Name, Class, Father Name, Address, Phone-No, Date of birth. This data which was stored repeatedly in file system in each application, need not be stored repeatedly, because every other application can access this information by joining of relations on the basis of common column i.e. Roll no. Suppose any user of Library system needs the Name and Address of any particular student, then by joining of Library and General Office relations on the basis of column Roll no, he\she can easily retrieve this information.

Thus we can say that centralized system of DBMS reduces the redundancy of data to a great extent but cannot eliminate the redundancy because Roll no is still repeated in all the relations.

2. Integrity can be enforced

Integrity of data means that data in the database is always accurate, that is incorrect information cannot be stored in database. In order to maintain the integrity of data, some integrity constraints are enforced on the database. A DBMS should provide capabilities for defining and enforcing the constraints.

For Example: Let us consider the case of college database and suppose that college is having only BA,BCA, BIT, BBA and BCOM classes. But if a user enters the class -CA, then this incorrect information must not be stored in the database and must be prompted that this is an invalid data entry . In order to enforce this, the integrity constraints must be applied to the class attribute of the student entity. But, in case of file system this constraint must be enforced on all the applications separately (because all applications have class field).

In case of DBMS, this integrity constraint is applied only once on the class field of the General Office (because class field appears only once in the whole database), and all other applications will get the class information about the student from the General Office table so the integrity constraint is applied to the whole database. Now, we can say that integrity constraint is applied to the whole database. Therefore, we can say that integrity constraint can be easily enforced in centralized DBMS system as compared to file system.

3. Inconsistency can be avoided

When the same data is duplicated and changes are made at one site, which is not propagated to the other site, it gives rise to inconsistency and the two entries regarding the same data will not agree. At such times the data is said to be inconsistent. So if the redundancy is removed chances of having inconsistent data is also removed.

Let us again consider the college system and suppose that in case of General Office file it is indicated that Roll-no 5 lives in Amritsar but in library file it is indicated that Roll-Number 5 lives in Jalandhar. Then this is a state at which the two entries of the same object do not agree with each other (that is one is updated and other is not). At such time the database is said to be inconsistent.

An inconsistent database is capable of supplying incorrect or conflicting information. So, there should be no inconsistency in database. It can be clearly shown that inconsistency can be avoided in centralized system very well as compared to file system.

Let us consider again the example of college system and suppose that RollNo5 is shifted from Amritsar to Jalandhar. The address information of Roll Number 5 must be updated, where ever Roll number and address occurs in the system. In case of file system, the information must be updated separately in each application, but if we make updation only at three places and forget to make updation at fourth application, the whole system shows the inconsistent results about Roll Number 5.

In case of DBMS, Roll number and address occur together only once in

General-Office table. So, it needs single updation and then all other applications retrieve the address information from General-Office which is updated. So, all applications will get current and latest information by providing single update operation and this single update operation is propagated to the whole database all other application automatically. This property is called as Propagation of Update.

We can say that the redundancy of data greatly affects the consistency of data. If redundancy is less, it is easy to maintain consistency of data. Thus DBMS system can avoid inconsistency to a great extent.

4. Data can be shared

As explained earlier, the data about Name, Class, Father-name etc. of General-Office is shared by multiple applications in DBMS as compared to file system. So now application s can be developed to operate against the same stored data. The applications may be developed without having to create any new stored files.

5. Standards can be enforced

Since DBMS is a central system, so standards can be enforced easily may be at Company level, Department level, National level or International level. The file system is an independent system so standards cannot be easily enforced on multiple independent applications.

6. Restricting unauthorized access

When multiple users share a database, it is likely that some users will not be authorized to access all information in the database. For example account office data is often considered confidential, and hence only authorized persons are allowed to access such data. In addition, some users may be permitted only to retrieve data, whereas other is allowed both to retrieve and to update. Hence the type of access operation retrieval or update must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions. The DBMS should then enforce these restrictions automatically.

7. Solving enterprise requirement than individual requirement

Since many types of users with varying level of technical knowledge use a database, a DBMS should provide a virility of user interface. The overall requirements of the enterprise are more important than the individual user requirements. So the DBA can structure the database system to provide an overall service that is “best for the enterprise”.

For example: A representation can be chosen for the data in storage that gives fast access for the most important application but at the cost of poor performance in some other application. But the file system favors the individual requirements than the enterprise requirements.

8. Providing Backup and Recovery

A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the program started executing.

1.2.5 Disadvantages of DBMS

The disadvantages of the database approach are summarized as follows:

1. Complexity

The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.

2. Size

The complexity and breadth of functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

3. Performance

Typically, a File Based system is written for a specific application, such as invoicing. As result, performance is generally very good. However, the DBMS is written to be more general, to cater for many applications rather than just one. The effect is that some applications may not run as fast as they used to.

4. Higher impact of a failure

The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of any component can bring operations to a halt.

5. Cost of DBMS

The cost of DBMS varies significantly, depending on the environment and functionality provided. There is also the recurrent annual maintenance cost.

6. Additional Hardware costs

The disk storage requirement for the DBMS and the database may necessitate the purchase of additional storage space. Furthermore, to achieve the required performance it may be necessary to purchase a larger machine, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

7. Cost of Conversion

In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications run on the new DBMS and hardware. This cost also includes the cost of training staff to use

these new systems and possibly the employment of specialist staff to help with conversion and running of the system. This cost is one of the main reasons why some organizations feel tied to their current systems and cannot switch to modern database technology.

1.2.6 Summary:

DBMS is basically a collection of programs that enables users to store, modify, and extract information from a database as per the requirements. DBMS is an intermediate layer between programs and the data. Programs range from small systems that run on personal computers to huge systems that runs on mainframes. DBMS must be able to accept data definitions (external schemas, the conceptual schemas, and all associated mappings) in source form and convert them to the appropriate object form. The DBMS must include a data manipulation language (DML) processor component. It must enforce certain recovery and concurrency controls. It must provide a data dictionary function.

1.2.7 Self Understanding:

Q1. Define DBMS?

Q2. What are the advantages of DBMS over traditional File Processing System

Q3. Explain the Characteristics of DBMS.

Q4. What are disadvantages of DBMS?

1.2.8 Further Readings:

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

DATABASE MANAGEMENT SYSTEM-II

Structure:

1.3.0 Introduction:

The database administrator is the person who implements the strategic and policy decisions regarding the data of the enterprise. Thus DBA is responsible for the overall control of the system at a technical level. The overall structure of the database is called database schema i.e. it is the description of a database, which is specified during the database design and is not expected to change frequently. The actual data in a database may change quite frequently. The data in the database at a particular moment of time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database. In the given database state, each schema construct has its own current set of instances.

1.3.1 Objectives

After completing this lesson, you will be able to:

- Explain the Implications of Database Approach
- Tell about the users of the database
- About the DBA and its responsibilities
- About Database Schema and instance

1.3.2 Implications of the Database Approach

Potential for Enforcing Standards. The database approach permits the DBA to define and enforce standards among database users in a large organization. This facilitates communication and cooperation among various departments, projects, and users within the organization. Standards can be defined for names and formats of data elements, display formats, report structures, terminology, and so on. The DBA can enforce standards in a centralized database environment more easily than in an environment where each user group has control of its own files and software.

Reduced Application Development Time. A prime selling feature of the database approach is that developing a new application such as the retrieval of certain data from the database for printing a new report-takes very little time. Designing and implementing a new database from scratch may take more time than writing a single specialized file application . However, once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities. Development time using a DBMS is estimated to be one-sixth to one-fourth of that for a traditional file system.

Flexibility: It may be necessary to change the structure of a database as requirements change, For example, a new user group may emerge that needs information not currently in the database. In response, it may be necessary to add a file to the database or to extend the data elements in an existing file. Modern DBMSs allow certain types of changes to the structure of the database without affecting the stored data and the existing application programs.

Availability of Up-to-Date Information: A DBMS makes the database available to all users. As soon as one user's update is applied to the database, all other users can immediate see this update. The availability of up-to-date information is essential for many transaction-processing applications, such as reservation systems of a DBMS.

Economies of Scale: The DBMS approach permits consolidation of data and applications, thus reducing the amount of wasteful overlap between activities of data-processing personnel in different projects or departments. This enables the whole organization to invest in more powerful processors, storage devices or communication gear, rather than having each department purchase its own (weaker)equipment. This reduces overall costs of operation and management.

1.3.3 User of Database:

There are a number of users who can access or retrieve data on demand using application programs and interfaces provided by DBMS. Each type of user needs different software capabilities. Following are the categories of the users:

1. Application Programmers
2. End Users
3. Database Administrator (DBA)

Application Programmers: Users who are responsible for writing application programs that use database. These programs operate on data for retrieving existing information, inserting new information, deleting or changing existing information.

End Users: End Users are those who need not know about the presence of database system or any other system supporting their usage. These users interact with the system via the interface (menu- or- form driven) provided by DBMS. For example: Users of Automatic Teller machines (ATM) fall under this category.

Database Administrators (DBA): The database administrator is the person who implements the strategic and policy decisions regarding the data of the

enterprise. Thus DBA is responsible for the overall control of the system at a technical level. Detailed responsibilities of DBA will be discussed in the following section.

1.3.4 DBA and its responsibilities

The data administrator is the person who makes the strategic and policy decisions regarding the data of the enterprise. Database Designer or Data Administration is the person responsible for indentifying the data to be stored in the data base and for choosing appropriaate structures to represent and store this data. It is the responsibility of the data base designer to communicate with all the prospective users in order to understand their requirements and to create a design that meets these requirements. And the Data Base Administrator (DBA) is the person who provides the necessary technical support for implementing the decisions taken by Data Administrator. Thus, the DBA is responsible for the overall control of the system at a technical level. In general, those functions will include the following.

Defining the conceptual schema

It is the data administrator's job to decide exactly what information is to be held in the database-in other words, to identify the entities of interest to the enterprise and to identify the information to be recorded about those entities. This process is usually referred to as logical-sometimes conceptual-database design. Once the data administrator has thus decided the content of the database at an abstract level, the DBA will then create the corresponding conceptual schema, using the conceptual DDL. The object (compiled) form of that schema will be used by the DBMS in responding to access requests. The source (uncompiled) form will act as a reference document for the users of the system.

Defining the internal schema

The DBA must also decide how the data is to be represented in the stored database. This process is usually referred to as physical database design. Having done the physical design, the DBA must then create the corresponding storage structure definition (i.e., the internal schema), using the internal DDL. In addition, he or she must also define the associated mapping between the internal and conceptual schemas. In practice either the conceptual DDL, or the internal DDL-most likely the former-will probably include the means for defining that mapping, but the two functions (creating the schema, defining the mapping) should be clearly separable. Like the conceptual schema, the internal schema and corresponding mapping will exist in both source and object form.

Liaising with users

It is the business of the DBA to liaise with users, to ensure that the data they require is available, and to write (or help the users write) the necessary external schemas, using the applicable external DDL. In addition, the mapping between any given external schema and the conceptual schema must also be defined. In practice, the external DDL will probably include the means for specifying that mapping should

be clearly separable . Each external schema and corresponding mapping will exist in both source and object form.

Other aspects of the user liaison function include consulting on application design, providing technical education, assisting with problem determination and resolution, and similar system-related professional services.

Defining security and integrity procedures

The conceptual DDL should include facilities for specifying security and integrity rules that can be regarded as part of the conceptual schema**.**

Defining backup and recovery procedures

Once an enterprise is committed to a database system, it becomes critically dependent on the successful operation of that system. In the event of damage to any portion of the database-caused by human error, say, or a failure in the hardware or supporting operating system-it is essential to be able to repair the data concerned with the minimum of delay and with as little effect as possible on the rest of the system. For example, the availability of data that has not been damaged should ideally not be affected. The DBA must define and implement an appropriate recovery scheme, involving, e.g., periodic unloading "dumping" of the database to backup storage, and procedures for reloading the database when necessary from the most recent dump.

Incidentally, the foregoing discussion provides one reason why it might be a good idea to spread the total data collection across several databases, instead of keeping it all in one place, the individual database might very well form the unit for dump and reload purposes. Nevertheless, we will continue to talk as if there were in fact just a single database, for simplicity.

Monitoring performance and responding to changing requirements

The DBA is responsible for so organizing the system as to get the performance that is "best for the enterprise" and for making the appropriate adjustments as requirements change. For example, it might be necessary to reorganize the stored database on a periodic basis to ensure that performance levels remain acceptable. As already mentioned, any change to the physical storage (internal) level of the system must be accompanied by a corresponding change to the definition of the mapping from the conceptual level, so that the conceptual schema can remain constant.

Of course, the foregoing is not an exhaustive list it is merely intended to give some idea Of the extent and nature of the DBA's responsibilities.

1.3.5 Database Schema and Instance:

The overall structure of the database is called database schema i.e it is the description of a database, which is specified during the database design and is not expected to change frequently. The diagram that displays the structure of each record type but not the actual instance of records is called a schema diagram. A schema

diagram displays only some aspects of a schema, such as names of the records types and data items, and some types of constraints. For example:

University_Dept:

Dept_Id Dept_name Date_Of_Estb Head

Student_Record

Stu_Id Stu_Name D_O_Adm Class Session

Employee_Record

Emp_ID Emp_Name Emp_Add Dept D_O_J

The actual data in a database may change quite frequently. For Example: the database shown above changes every time we add a student. The data in the database at a particular moment of time is called a database state or snapshot. It is also called the current set of occurrences **or instances in the database**. In the given database state, each schema construct has its own current set of instances. For example, the Student_Record construct will contain the set of individual student records as its instances. Every time we insert or delete a record, or change the value of a data item in a record, we change one state of the database into another state.

The distinction between database schema and database state is very important. When we define a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database state is first populated or loaded with the initial data. From then on, every time an update operation is applied to the database, we get another database state. At any point in time, the database has a current state. The DBMS is partly responsible for ensuring that every state of the database has a valid state i.e. a state that satisfies the structure and constraints specified in the schema. Hence, specifying a correct schema to the DBMS is extremely important, and the schema must be designed with the utmost care. The DBMS stores the description of the schema constructs and constraints – also called the Meta-data- in the DBMS catalog so that the DBMS software can refer to the schema whenever it needs to. The schema is sometimes called the intension and database state is called the extension of the schema. As we have already mentioned, the database schema do not change frequently, but in case there is a need to add a data item to the record type of database, this is known as schema evolution. For example – We need to add data item Emp_DOB to the record type Employee_Record.

1.3.6 Summary:

There are a number of users who can access or retrieve data on demand using application programs and interfaces provided by DBMS. Each type of user needs different software capabilities. Following are the categories of the users: Application Programmers, End Users, Database Administrator (DBA). The data administrator is the person who makes the strategic and policy decisions regarding the data of the

enterprise. And the database administrator (DBA) is the person who provides the necessary technical support for implementing those decisions. Thus, the DBA is responsible for the overall control of the system at a technical level. In general, the functions of DBA are Defining the conceptual schema, Defining the internal schema, Liaising with users, Defining security and integrity procedures, Monitoring performance and responding to changing requirements and lot more.

1.3.7 Self Understanding:

Q1. Explain the Implications of Database approach.

Q2. Explain the users of database.

Q3. Define DBA and explain its responsibilities.

Q4. What do you mean by Database Schema ands instance?

Q5. Differentiate between Data Administrator and Data Base Administrator.

1.3.8 Further Readings:

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

DBMS ARCHITECTURE

Structure:

1.4.0 Introduction:

The Three Levels of the Architecture of DBMS is also known as ANSI/SPARC Model. The goal of this three level architecture is to separate the user applications and the physical database. The ANSI/SPARC model is divided into three levels, known as the internal, conceptual, and external levels. In a DBMS based on three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. The process of transforming requests and results between levels are called mappings. There are two levels of mappings in the architecture – Conceptual/Internal mapping and External/Conceptual mapping. The three level architecture is then used to explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. There are two types of data independence – Logical data Independence and Physical data independence.

1.4.1 Objectives

After completing this lesson, you will be able to:

- Three Level DBMS Architecture or ANSI/SPARC Model
- Data Independence
- Mappings between Different levels of DBMS Architecture

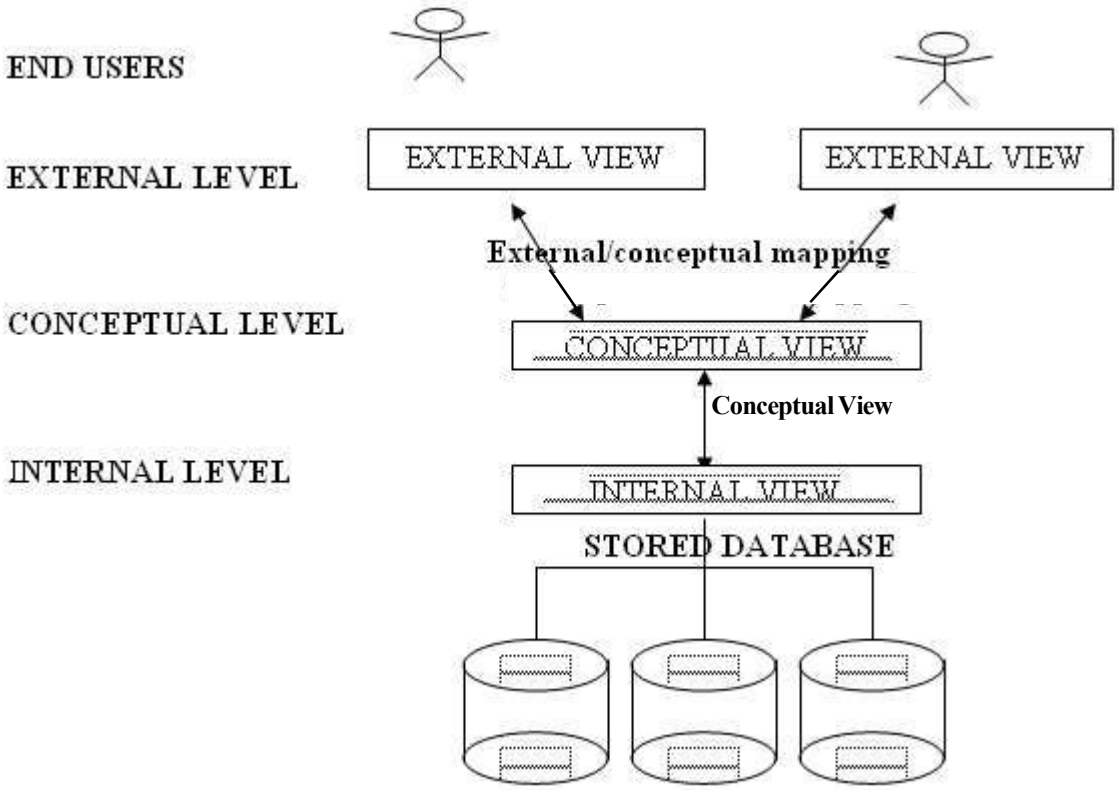
1.4.2 Three-Level Architecture of DBMS or ANSI/SPARC Model:

Several different frameworks have been suggested over the last several years. For example, a framework may be developed based on the functions that the various components of a DBMS must provide to its users. It may also be based on different

views of data that are possible within a DBMS. Commonly used views of data approach is the three-level architecture suggested by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee). ANSI/ SPARC produced an interim report in 1972 followed by a final report in 1977. The reports proposed an architectural framework for databases. A large number of commercial systems and research database models fit into this framework. The three levels of the architecture are three different views of the data:

1. *External* - individual user view
2. *Conceptual* - community user view
3. *Internal* - physical or storage view

The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout. A database system that is able to separate the three different views of data is likely to be flexible and adaptable. This flexibility



The view at each of these levels is described by a scheme. **A Scheme** is an outline or a plan that describes the records and relationships existing in the view. The word scheme, which means a systematic plan for attaining some goal, is used

interchangeably in the database literature for the plural instead of schema. The word schema is used in the database literature for the plural instead of schemata, the grammatically correct word. The scheme also describes the way in which entities at one level of abstraction can be mapped to the next level.

External or User View

The external or user view is at the highest level of database abstraction where only those portions of the database that are of concern to a user or application program are included. Any number of user views (some of which may be identical) may exist for a given global or conceptual view.

Each external view is described by means of a scheme called an **external schema.** The external schema consists of the definition of the logical records and the relationships in the external view. The external schema also contains the method of deriving the objects in the external view from the objects in the conceptual view. The objects include entities, attributes, and relationships. (The terms view, scheme, and schema are sometimes used interchangeably when there is no confusion as to what is implied.)

Conceptual or Global View

At this level of database abstraction all the database entities and the relationships among them are included. One conceptual view represents the entire database. This conceptual view is defined by the **conceptual schema**. It describes all the records and relationships included in the conceptual view and, therefore, in the database. There is only one conceptual schema per database. This schema also contains the method of deriving the objects in the conceptual view from the objects in the internal view.

The description of data at this level is in a format independent of its physical representation. It also includes features that specify the checks to retain data consistency and integrity.

Internal View

It is at the lowest level of abstraction, closest to the physical storage method used. It indicates how the data will be stored and describes the data structures and access methods to be used by the database. The internal view is expressed by the **internal schema,** which contains the definition of the stored record, the method of representing the data fields, and the access aids used. At least the following aspects are considered at this level:

1. Storage allocation e.g. B-trees, hashing etc.
2. Access paths e.g. specification of primary and secondary keys, indexes and pointers and sequencing.
3. Miscellaneous e.g. data compression and encryption techniques, optimization of the internal structures.

Efficiency considerations are the most important at this level and the data

structures are chosen to provide an efficient database. The internal view does not deal with the physical devices directly. Instead it views a physical device as a collection of physical pages and allocates space in terms of logical pages.

The separation of the conceptual view from the internal view enables us to provide a logical description of the database without the need to specify physical structures. This is often called *physical data independence*. Separating the external views from the conceptual view enables us to change the conceptual view without affecting the external views. This separation is sometimes called *logical data independence*.

Assuming the three level view of the database, a number of mappings are needed to enable the users working with one of the external views. For example, the payroll office may have an external view of the database that consists of the following information only:

1. Staff number, name and address.
2. Staff tax information e.g. number of dependents.
3. Staff bank information where salary is deposited.
4. Staff employment status, salary level, leave information etc.

The conceptual view of the database may contain academic staff, general staff, casual staff etc. It will be needed to create a mapping where all the staff in the different categories are combined into one category for the payroll office. The conceptual view would include information about each staff's position, the date employment started, full-time or part-time, etc. This will need to be mapped to the salary level for the salary office. Also, if there is some change in the conceptual view, the external view can stay the same if the mapping is changed.

1.4.3 Data Independence:

For simplifying the concept of data independence, let us explain the opposite of data independence. Applications implemented on older systems tend to be data dependent. **Data dependent** means the way in which the data is organized in secondary storage, and the technique for accessing it, are both dictated by the requirements of the application under consideration and moreover that knowledge of that data organization and access technique is built into the application logic and code but on the other hand, in a database system, data dependence is extremely undesirable, at least for following two reasons:-

1. Different applications will need different views of the same data.
2. The DBA must have the freedom to change the storage structure or access technique in response to changing requirements, without having to modify existing applications.

Thus, **Data Independence can be defined as** the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. There are two types of data independences:-

1. Logical data Independence.
2. Physical data Independence.

Difference between Logical Data Independence and Physical Data Independence:

1. Logical data independence is usually required for changing the conceptual schema without having to change the external schema or application programs while physical data independence is required for changing the internal schema without having to change the external schema or conceptual schema
2. Logical data independence specifies that the application programs need not to be changed if new fields are added to, or deleted from a database. It is concerned with changing the logical structure without having to change the application programs while physical data independence is concerned with changing the internal schema i.e. changing the physical storage of the data without having to change the corresponding external schema.
3. Logical data Independence is usually more difficult to achieve than physical data independence because the programs required to retrieve the data are heavily dependent on the logical structure of data while physical data independence is comparatively easy to achieve than the logical data Independence.
4. Logical data independence is concerned with changing the structure of the data or changing the data definition while physical data independence helps to migrate the files from one kind of storage device to another.

Advantages of Data Independence:

1. It provides the capacity to change the schema at one level without having the need to change the schema at the next higher level.
2. It also hides the implementation details or hardware level details from its users so that the users can concentrate on the program only.
3. The various changes made at different levels are absorbed by mapping at different level.
4, Physical data independence allows the changes to be made in storage of data without affecting application programs.
5. Due to logical data independence, there is no effect to application programs even if new fields are added or old records are deleted from the existing data.
6. Physical data independence allows the files to migrate from one kind of storage device to another.
7. Logical data independence does not require the external schema to be changed.
8. Physical data independence does not require the internal schema to be changed.

1.4.4 Mapping between Different Levels:

The conceptual **database** is the model or abstraction of the objects of concern to the database. Thus, the conceptual record of above Figure is the conceptual database and represents the abstraction of all the applications involving the entity set EMPLOYEE. The view is the subset of the objects modeled in the conceptual database that is used by an application. These could be any number of views of a conceptual database. A view can be used to limit the portion of the database that is known and accessible to a given application.

Two mappings are required in a database system with three different views. A mapping between the external and conceptual views gives the correspondence among the records and the relationship of the external and conceptual views. The external view is an abstraction of the conceptual view, which in its turn is an abstraction of the internal view. The user of the external view sees and manipulates a record corresponding to the external view. There is a mapping from a particular logical record in the external view to one (or more) conceptual record(s) in the conceptual view. A number of differences could exist between the two. Names of the fields and records, for instance, may be different. A number of conceptual fields can be combined into a single logical field, for example, Last_Name and First_Name at the conceptual level but Name at the logical level. A given logical record could be derived from a number of conceptual records.

Similarly, there is a mapping from a conceptual record to an internal one. An internal record is a record at the internal level, not necessarily a stored record on a physical storage device. The internal record of above figure may be split up into two or more physical records. The **physical database** is the data that is stored on secondary storage devices. It is made up of records with certain data structures and organized in files. Consequently, there is an additional mapping from the internal record to one or more stored records on secondary storage devices. This may have been implemented using some form of nonlinear addressing. The internal record is assumed to be linearly addressed. However, this complexity is managed by the DBMS and the user need not be aware of its presence nor be concerned with it.

Mapping between the conceptual and the internal level specifies the method of deriving the conceptual record from the physical database. Again, differences similar to those that exist between external and conceptual views could exist between the conceptual and internal views. Such differences are indicated and resolved in the mapping.

Differences that could exist, besides the difference in names, include the following;

- Representation of numeric values could be different in the two views. One view could consider a field to be decimal, whereas the other view may regard

the field as binary. A two-way transformation between such values can be easily incorporated in the mapping. If, however, the values are stored in a binary format, the range of values may be limited by the underlying hardware.

- Representation of string data can be considered by the two views to be coded differently. One view may perceive the string data to be in ASCII code, the other view may consider the data to be in EBCDIC code. Again, two-way transformation can be provided.
- The value for a field in one view could be computed from the values in one or more fields of the other view. For example, the external view may use a field containing a person's age, whereas the conceptual view contains the date of birth. The age value could be derived from the date of birth by using a date function available from the operating system. Another example of a computed field would be where an external view requires the value of the hours worked during a week in a field, whereas the conceptual view contains fields representing the hours worked each day of the week. The former can be derived from the later by simple addition. These two examples of transformation between the external and conceptual views are not bidirectional. One cannot uniquely reflect a change in the total hours worked during a week to hours worked during each day of the week. Therefore, a user's attempt to modify the corresponding external fields will not be allowed by the DBMS.

Such mapping between the conceptual and internal levels is a correspondence that indicates how each conceptual record is to be stored and the characteristics and size of each field of the record. Changing the storage structure of the record involves changing the conceptual view to internal view mapping so that the conceptual view does not require any alteration.

The conceptual view can assume that the database contains a sequence of records for each conceptual record type. These records could be accessed sequentially or randomly. The actual storage could have been done to optimize performance. A conceptual record may be split into two records, with the less frequently used record (part of the original record) on a slower storage device and the more frequently used, record, on a faster device. The stored record could be in a physical sequence, or one or more indices may be implemented for faster access to record occurrences by the index fields. Pointers may exist in the physical records to access the next record occurrence in various orders. These structures are hidden from the conceptual view by the mapping between the two.

1.4.5 Summary:

The three level architecture is divided into three levels: the external level, the conceptual level, and the internal level. The external or user view is at the highest level of database abstraction where only those portions of the database of

concern to a user or application program are included. One conceptual view represents the entire database. The conceptual view is defined by the **conceptual schema**. It describes all the records and relationships included in the conceptual view and, therefore, in the database. The internal view indicates how the data will be scored and describes the data structures and access methods to be used by the database. It is expressed by the **internal schema,** which contains the definition of the stored record, the method of representing the data fields, and the access aids used. The DBMS provides users with a method of abstracting their data requirements and removes the drudgery of specifying the details of the storage and maintenance of data. The DBMS insulates users from changes that occur in the database. Two levels of database independence are provided by the system. Physical independence allows changes in the physical level of data storage without affecting the conceptual view. Logical independence allows the conceptual view to be changed without affecting the external view.

1.4.6 Self Understanding:

Q1. Define Data Independence.

Q2. Explain the difference between Logical data Independence and Physical data Indepence.

Q3. Explain ANSI/SPARC Model or The Three Level Architecture or DBMS.

Q4. Explain the mapping between different levels of DBMS Architecture.

Q5. What do you mean by data abstraction?

1.4.7 Further Readings:

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

DATABASE LANGUAGES

Structure:

1.5.0 Introduction:

There is variety of users supported by a DBMS. Thus, DBMS must provide appropriate languages and interfaces for each category of users. Once the design of a database is completed and a DBMS is chosen to implement the database, the first order of the day is to specify conceptual and internal schema for the database and any mappings between the two. In many DBMSs where no strict separation of levels is maintained, one language, called the Data Definition Language (DDL), is used by the DBA and by database designers to define both schemas. Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion and modification of the data. The DBMS provides a data manipulation language (DML) for these purposes. DBMS also provides a language named as Data Control Language (DCL) for granting and revoking the privileges for using the database.

A Key is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value of the key attributes at the same time. There are different type of keys – Super, Candidate, Primary, Unique and Foreign. Keys help in maintaining the data integrity and consistency.

1.5.1 Objectives

After completing this lesson, you will be able to:

- What is DBMS Language?
- DDL, DML, DCL
- Keys

1.5.2 Database Languages

There is variety of users supported by a DBMS. Thus, DBMS must provide appropriate languages and interfaces for each category of users.

Following are the languages provided for various categories of users for different operations:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Control Language (DCL)

1.5.2.1 Data Definition Language

Once the design of a database is completed and a DBMS is chosen to implement the database, the first order of the day is to specify conceptual and internal schema for the database and any mappings between the two. In many DBMSs where no strict separation of levels is maintained, one language, called the Data Definition Language (DDL), is used by the DBA and by database designers to define both schemas. The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog. DDL statements are used to define, alter, or drop database objects. The following table gives an overview about the usage of DDL statements;

S.No.	Purpose	SQL DDL Statement
1.	Create database schema objects	Create
2.	Alter database schema objects	Alter
3.	Delete database Schema objects	Drop
4.	Rename schema objects	Rename

1.5.2.2 Data Manipulation Language

Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion and modification of the data. The DBMS provides a Data Manipulation Language (DML) for these purposes. There are two main types of DMLs - A high-level or nonprocedural DML and a low-level or procedural DML. A high-level or nonprocedural DML can be used on its own to specify complex database operations in a concise manner. Many DBMSs allow high-level DML statements either to be entered interactively from a terminal or to be embedded in a general-purpose programming language. In the latter case, DML statements must be identified within the program so that they can be extracted by a pre-compiler and processed by the DBMS. A low-level or procedural DML must be embedded in a general-purpose programming language. This type of DML typically retrieves individual records or objects from the database and processes each separately. Hence, it needs to use programming language constructs, such as looping, to retrieve and process each record from a set of records. Low-level DMLs

are also called record-at-a-time DMLs because of this property. High-level DMLs, such as SQL, can specify and retrieve many records in asingle DML statement and are hence called set-at-a-time or set-oriented DMLs. A query in a high-level DML often specifies which data to retrieve rather than how to retrieve it; hence, such languages are also called declarative.

The following are basic SQL commands that used as DML statements:

S.No.	Purpose	SQL statement
1.	Retrieve data from one or more tables	Select
2.	Add new rows in a table	Insert
3.	Remove rows from a table	Delete
4.	Change data in rows in a table	Update

5.2.3 Data Control Language

DBMS also provides a language named as Data Control Language (DCL) for granting and revoking the privileges for using the database. A privilege can either be granted to a user with the help of GRANT statement. The privileges assigned can be Select, Alter, Delete, Execute, Insert, Index etc. In addition to granting of privileges, you can also revoke it by using Revoke command. Following are the SQL Statements that can be used as DCL:

S.No.	Purpose	SQL statement
1.	Granting the privileges	Grant
2.	Taking away the privileges	Revoke
3.	Adding a comment to the data dictionary	Comment

1.5.3 Keys

Before we start discussing the concept of Key and various types of keys - Consider the Table PGDCA_Students_Details (Roll_No, SSNo, Student_Name, Father_name, Address, Phone_No). Here in this table, PGDCA_Students_Details is the name of the table and (Roll_No, SSNo, Student_Name, Father_name, Address, Phone_No) are the names of the attributes (fields) of the table. It is storing the details of the students of the PGDCA class.

Table is also called entity set and the rows in the table are also called entities.

A key is a single attribute or combination of two or more attributes of an entity set that is used to identify one or more instances of the set. The attribute Roll_No uniquely identifies an instance of the entity set PGDCA_Students_Details. Thus, a key is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value of the key attributes at the same time.

There are various types of keys that are as follows:

1. Super Key
2. Candidate Key

3. Primary Key
4. Alternative Key
5. Unique Key
6. Foreign Key

Details of above keys will be discussed in chapter 6 .

1.5.4 Summary

DBMS must provide appropriate languages for various categories of DBMS users. After database design, a DBMS is chosen to implement the database. Most common DBMS languages are DDL, DML, and DCL. Data Definition Language (DDL) is used by the DBA and by database designers to define database schemas. The DBMS provides a Data Manipulation Language (DML) for manipulating data in database. Data Control Language (DCL) is used for granting and revoking the privileges to the users for using the database.

A key is a single attribute or combination of two or more attributes of an entity set that is used to identify one or more instances of the set uniquely. There is different type of keys – Super, Candidate, Primary, Unique and Foreign. Keys help in maintaining the data integrity and consistency.

1.5.5 Self Understanding

Q1. What do you mean by DBMS Language?
Q2. List the various Database Languages? Explain each of them.
Q3. Write short notes on DDL, DML, DCL.
Q4. What are types of DML? Explain each of them.
Q5. What do you mean by key? List down the various keys available.

1.5.6 Further Readings

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

DATA MODELS AND KEYS

Structure:

1.6.0 Introduction:

DBMSs have database utilities that help the DBA in managing the database system. Common utilities include loading, backup, file reorganization and performance monitoring. One fundamental characteristic of the database approach is that it provides some level of data abstraction by hiding details of data storage that are not needed by most database users. A data model is a collection of concepts that can be used to describe the structure of a database and that provides the necessary means to achieve the data abstraction. We can categorize the data models into High-level or conceptual data models and Low-level or physical data models. High-level or conceptual data models are those that provide concepts that are close to the way many users perceive data. Entity Relationship model is a popular example of high-level data model. Low-level or physical data models are those that provide concepts that describe the details of how data is stored in the computer. Between the two broad categories of data models is representational or implementation data model which provide concepts that may be understood by end users. Hierarchical, Network and relational are the popular examples of this category of data model. A Key is a property of the entity set, rather than of the individual entities. Any two

individual entities in the set are prohibited from having the same value of the key attributes at the same time. There are different type of keys – Super, Candidate, Primary, Unique and Foreign. Keys help in maintaining the data integrity and consistency.

1.6.1 Objectives

After completing this lesson, you will be able to:

- Explain the database utilities
- Define Data Model and various data models
- Define Key
- Explain different type of keys – Super, Candidate, Primary, Unique and Foreign

1.6.2 Database Utilities

DBMSs have database utilities that help the DBA in managing the database system. Common utilities have the following types of functions:

1. ***Loading:*** A loading utility is used to load existing data files — such as text files or sequential files—into the database. Usually, the current format of the data file and the desired database file structure are specified to the utility, which then automatically reformats the data and stores it in the database. With the proliferation of DBMSs, transferring data from one DBMS to another is becoming common in many organizations. Some vendors are offering products that generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas). Such tools are also called conversion tools.
2. ***Backup:*** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape. The backup copy can be used to restore the database in case of catastrophic failure. Incremental backups are also often used, where only changes since the previous backup are recorded. Incremental backup is more complex but it saves space.
3. ***File reorganization:*** This utility can be used to reorganize a database file into a different file organization to improve performance.
4. ***Performance monitoring:*** Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files to improve performance.

   Other utilities may be available for sorting files, handling data compression, monitoring access by users, and performing other functions.

1.6.3 Data Models:

A data model is a collection of concepts that can be used to describe the structure of a database and that provides the necessary means to achieve the data abstraction. A Data Model is a collection of concepts that can be used to describe the structure of the database including data types, relationship and constraints

that should apply on the data.

The main purpose of data modeling is to help in understanding the meaning of the data and to facilitate communication about information requirements. A data model supports communication between the users and database designers.

WHY DATA MODELLING IS IMPORTANT?

The goal of data model is to make sure that all data objects required by the database are completely and accurately represented because the data model uses easily understood notations and natural language which can be reviewed and verified as correct by the end users.

The data model must be detailed enough to be used by the database designers for building the physical database. The information contained in data model will be used to define the relational tables, primary and foreign keys and other information. A poorly designed database will require more time in the long term. Without careful planning you may create a database that omits data required to create reports, produce results that are incorrect or inconsistent and is unable to accommodate changes in the user's requirements. A data model should possess the following characteristics so that the best possible data representation can be obtained:

- Diagrammatic representation of the Data Model
- Simplicity in designing and expressibility to distinguish between data and their relationships. Also it is detailed enough to be used by a database designer to build the database
- Application independent i.e. it could be shared by different applications
- No duplication in representation of data
- Consistency and structure validity is maintained.

It follows a bottom up approach. A basic model, representing entities and relationships is developed first, and then details are added.

We can categorize the data models into three broad categories:

1.6.3.1 **High-level or conceptual data models:** High-level or conceptual data models are those that provide concepts that are close to the way many users perceive data. This category is also known as Object based data models. Most of the common data models under this category are:

a. **Entity Relationship Model (ER Model):** The ER Model is a high level conceptual data model which describes the structure of the database and the associated operations like retrieval and updation on the database. The basic concepts of ER Model include entity type, their attributes and relationships. It also represents certain constraints which are used on the database.

b. **Object Oriented Model:** The object oriented model is based on

collection of objects, attributes and relationships which together form the static properties. It also consists of integrity rules over objects and dynamic properties such as operations or rules defining new database states. An object is a collection of data and methods. When different objects of same type are grouped together they form a class. This model is used basically for multimedia applications as well as data with complex relationships. The object model is represented graphically with object diagrams containing object classes. Classes are arranged into hierarchies sharing common structure and behavior and are associated with other classes.

c. Semantic Data Models: These models are used to express greater interdependencies among entities of interest. These interdependencies enable the models to present the semantic of the data in the databases. This class of data models is influenced by the work done by artificial intelligence researchers. Semantic data models are developed to organize and represent knowledge but not data. This type of data models is able to express greater interdependencies among entities of interest. Mainframe database are increasingly adopting semantic data models. Also its growth usage is also seen in PC's. In coming times database management system will be partially or fully intelligent.

d. Functional Data Model: The functional data model describes those aspects of a system concerned with transformations of values-functions, mappings, constraints and functional dependencies. The functional data model describes the computations within a system. It shows how output value in computation are derived from input values without regard for the order in which the values are computed. It also includes constraints among values. It consists of multiple data flow diagrams. Data flow diagrams show the dependencies between values and computation of output values from input values and functions, without regard for when (or if) the functions are executed. Traditional computing concepts such as expression trees are examples of functional models, as are less traditional concepts such as spreadsheets.

High level data models use concepts such as entities, attributes and relationships. An entity represents a real-world object or concept, such as an employee or a project, that is described in the database. An attribute represents some property of interest that further describes an entity, such as the employee's name or salary. A relationship among two or more entities represents an interaction among the entities. For example, a works-on relationship between an employee and a project. The object oriented data model not only extends the state of the object but also the actions that are associated with the object, that is, its behavior. The object is said to encapsulate both state and behavior.

1.6.3.2 Low-level or physical data models: Low-level or physical data models are those that provide concepts that describe the details of how data is stored in the computer. It represents information such as record formats, record ordering, and access paths. An access path is a structure that makes the search for particular database records efficient. One of the most important low-level data model is unifying data model.

1.6.3.3 Representational or implementation data models: Between the two broad categories of data models is representational or implementation data model which provide concepts that may be understood by end users. These are used in describing data at logical and view levels of the three level architecture of DBMS. In contrast to object based data models, these are used to specify the overall logical structure of the database and to provide a higher-level description of the implementation. The most popular representational data models are

a. Hierarchical data model
b. Network data model
c. Relational data model

Representational data models represents data by using record structures and hence are sometimes called record-based data models.

1.6.4 Key:

Before we start discussing the concept of Key and various types of keys - Consider the Table PGDCA_Students_Details (Roll_No, SSNo, Student_Name, Father_name, Address, Phone_No). Here in this table, PGDCA_Students_Details is the name of the table and (Roll_No, SSNo, Student_Name, Father_name, Address, Phone_No) are the names of the attributes (fields) of the table. It is storing the details of the students of the PGDCA class.

Table is also called entity set and the rows in the table are also called entities.

Definition: A key is a single attribute or combination of two or more attributes of an entity set that is used to identify one or more instances of the set. The attribute Roll_No uniquely identifies an instance of the entity set PGDCA_Students_Details. Thus, a key is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value of the key attributes at the same time.

There are various types of keys that are as follows:

1. Super Key
2. Candidate Key
3. Primary Key
4. Alternative Key
5. Unique Key
6. Foreign Key

1.6.4.1 Super Key

It is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. It has the property of Uniqueness and Reducibility. For Example: SK1{Roll_No}, SK2{SSNO}, SK3{Roll_No,SSNo}, SK3{SSNo,Father_Name}, SK4{Roll_No,Student_Name}, SK5{Roll_No,Student_Name,Father_Name} .....

1.6.4.2 Candidate Key

If K is the Super Key, then so is any superset of K. We are basically interested in Super keys for which no proper subset is a super key. Such minimal super keys are called candidate keys. It has the property of Uniqueness and Irreducibility.
For Example:CK1{Roll_No},CK2{SSNO}.

1.6.4.3 Primary Key

A candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set. Such a minimal super key or candidate key is known as Primary Key. It has the property of Uniqueness and Irreducibility.
For Example: PK1 {Roll_No}

1.6.4.4 Alternate Key

Rest of the Candidate keys that are not taken as primary key are alternate keys.
AK1 {SSNO}, AK2 {Roll_No, Father_Name}.....

1.6.4.5 Unique

It is same as primary key. The only difference is that Unique allows NULLs but Primary key does not allow NULL.

1.6.4.6 Foreign Key

Let R1 be a relation (table) in which PK1 has been assigned as primary key. Let R2 be another relation in which PK2 is the primary key but PK1 (From Relation R1) has been marked as Foreign Key. The value of PK1 in any row of relation R2 must match with the value of PK1 of any row in relation R1.

For Example: In relation PGDCA_Student_Records {Roll_No} is acting as Primary Key. We have another relation Fee_Records (Slip_ID, Roll_No, Fee_paid, Date_of_Payment), Slip_Id is acting as Primary Key and PK1 {Roll_No} is acting as Foreign Key.

Is it possible to have a foreign key when there is only one table in a database?

Yes. There can be a case in which referencing and referenced relations are same and not distinct. Consider the relation EMP such that EMP (emp-name, mgr-ID)

In the above relation, emp_id is primary key and mgr_id is foreign key. SO, EMP Referencing emp_ id of EMP relation itself. So, EMP relation is acting as referencing as will as referenced. Thus, from the above example we see that there is only one table i.e. EMP and this table has a foreign key i.e. mgr_id.

1.6.5 Summary

The main purpose of data modeling is to help in understanding the meaning of the data and to facilitate communication about information requirements. A data model supports communication between the users and database designers. The goal of a data model is to make sure that all data objects required by the database are completely and accurately represented because the data model uses easily understood notations and natural language which can be reviewed and verified as correct by the end users. A key is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value of the key attributes at the same time. There are different type of keys – Super, Candidate, Primary, Unique and Foreign.

1.6.6 Self Understanding

Q1. What do you mean by database utility? Explain the various database utilities.

Q2. What do you mean by data model? Explain in brief various data models.

Q3. Define key. Explain different types of keys along with suitable examples.

Q4. What is the difference between Unique and Primary key?

Q5. Define Foreign Key? Is it possible to have a foreign key when there is only one table in a database?

Q6. List down the characteristics of Primary Key.

1.6.7 Further Readings

1. Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.
2. C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.
3. Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

LESSON NO. 1.7 **AUTHOR : VISHAL GOYAL**

ENTITY RELATIONSHIP MODEL (PART-1)

Structure:

1.7.0 Introduction:

The Entity-Relationship Model is a high-level conceptual data model developed by Chem in 1976 to facilitate database design. The E-R Model is shown diagrammatically using E-R diagrams which represents the elements of the conceptual model that show the meanings and the relationships between those elements independent of any particular DBMS and implementation details. Cardinality of a relationship between entities is calculated by measuring how many instances of one entity are related to a single instance of another. One of the main limitations of the E-R Model is that it cannot express relationship among relationships. So to represent these relationships among relationships. We combine the entity sets and their relationship to form a higher level entity set. This process of combining entity sets and their relationships to form a high entity set so as to represent relationships among relationships is called Aggregation.

1.7.1 Objectives

After completing this lesson, you will be able to:

- Define E-R Model
- Explain the mapping Cardinalities
- Define and Draw E-R diagrams
- Define weak and strong entity sets
- Define aggregation

1.7.2 Entity Relationship Model

The entity-relationship model is based on the 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 Model is

extremely useful in mapping the meanings and interactions of real-world enterprises into a conceptual schema. Entity – relationship model was originally proposed by Peter in 1976 as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the entity relationship diagram, which is used to visually represent data objects. For the database designer, the utility of the ER model is :

1) It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
2) It is simple and easy to understand with a minimum of training. Therefore the model can be used by the database designer to communicate the design to the end user.
3) In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.

1.7.2.1 Basic Concepts

There are three basic notions that the E-R data model employs – entity sets, relationship sets, and attributes.

Entity : An entity is a thing or object in the real world that is distinguishable from all other objects. For example, each person in an enterprise is an entity. An entity has set of properties and the values for some set of properties may uniquely identify an entity. For example, the employee Id of a person uniquely identifies one particular person in the enterprise. Entities are principal data object about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract such as person, places, things, or events which have relavence to the database. Some specific examples of entities are EMPLOYEES, PROJECTS and INVOICES. An entity is analogous to a table in the relational model. Entities are classified as :

1) **Independent :** An independent entity is one that does not rely on another for identification.
2) **Dependent :** A dependent entity is one that relies on another for identification.

Special Entity Types : Associative entities (also known as intersection entities) are entities used to associate two or more entities in order to reconcile a many-to-many relationship. Subtypes entities are used in generalization hierarchies to represent a subset of instances of their parent entity, called the supertype, but which have attributes or relationships that apply only to the subset.

Entity set : An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all persons who are customers at a given bank, for example, can be defined as the entity set customer. The individual entities that constitute the set are said to be the extension of the entity set. Thus all the customers of the given bank are the extensions of the entity set customer.

Attributes : An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. For example, a

customer entity set of a given bank has the attributes like account number, customer name, customer address etc. For each attribute, there is a set of permitted values, called the domain or value set, of that attribute. The domain of attribute customer-name might be the set of all text strings of a certain length.

A database thus includes a collection of entity sets, each of which contains any number of entities of the same type. For example, a bank database consists of entity sets like customer, loan etc.

An attribute, as used, in the E-R Model, can be characterized by the following attribute types:

1. **Simple and composite attributes**: Simple attributes are those that cannot be divided into subparts i.e. into other attributes. For example: customer account number is a simple attribute. Composite attributes are those that can be further divided into subparts or attributes. For example: Customer_name attribute of an entity can be considered as composite attribute because it can further be divide into subparts like first name, middle name and last name.
2. **Single valued and multivalued attributes**: The attributes that have a single value for a particular entity is known as single valued attributes. For example, employee_Id of an employee in an enterprise will be single valued for every employee. Multivalued atributes are those attributes that have multiple values for an entity. For example, employee_dependent_names for a particular employee in an enterprise can have zero, one or more names depending on the number of dependents of an employee.
3. **Null attributes:** A null value is used when an entity does not have a value for an atribute. For example, if a particular employee has no dependents, then the value of employee_dependent_names for that employee in an enterprise will be null. Null can also designate that an attribute value is unknown. An unknown value may be either missing or not known.
4. **Derived attributes**: The value of this type of attributes is derived from the values of other related attributes or entities. Some attributes may be related for a particular entity For example: the age of an employee can be derived from the date_of_birth attribute of an employee therefore they are related .

Relationship Sets: A relationship is an association among several entities. For example, we can define a relationship that associates an employee Ram Singh with Department Computer Science. This relationship specifies that Employee Ram Singh is working in the department Computer Science.

A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation between n>=2 entity sets. If $E_1,E_2,E_3.....En$ are entity sets, then relationship set R is a subset of $\{(e_1,e_2,e_3.....e_n) \mid e_1 \varepsilon E_1, e_2 \varepsilon E_2, e_3 \varepsilon E_3..... e_n \varepsilon E_n\}$ where $(e_1,e_2,e_3.....e_n)$ is a relationship.

Consider the two entity sets *employee* and *Departments*. We define the relationship set *works for* to denote the association between employee and the department that the employees have.

The association between entity sets is referred to as participation, that is the entity sets $e_1 \varepsilon E_1$, $e_2 \varepsilon E_2$, $e_3 \varepsilon E_3$..... $e_n \varepsilon E_n$ participate in relationship set R. A relationship instance in an E-R schema represents that an association exists between the named entities in the real world enterprise that is being modeled.

The function that an entity plays in a relationship is called that entity's role. Since entity sets participating in a relationship set are generally distinct, roles are implicit and are not usually specified. However, they are useful when the meaning of a relationship needs clarification. Such is a case when the entity sets of a relationship set are not distinct, i.e., the same entity set participates in a relationship set more than once, in different roles. In this type of relationship set, which is sometimes called a recursive relationship set, explicit role names are necessary to specify how an entity participates in a relationship instance.

A relationship can also have descriptive attributes. Consider a relationship set *depositor* with entity set *customer* and *account*. We could associate the attribute *access-date* to that relation to specify the most recent date on which a customer accessed an account. The *depositor* relationship among the entities corresponding to customer is described by access-date, which means when customer has most recently accessed the account.

The number of entity sets that participate in a relationship set is also the **degree** of the relationship set. A binary relationship set is of degree 2; a ternary relationship set is of degree 3.

1.7.3 Mapping Cardinalities:

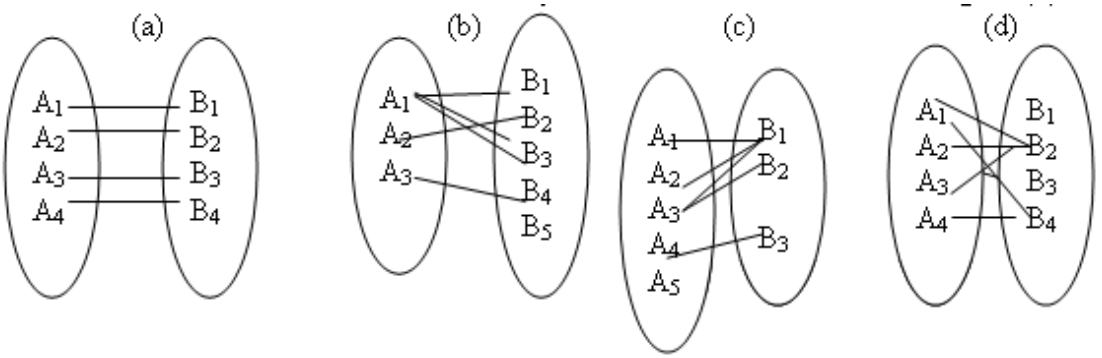
Mapping Cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationship sets, although occasionally they contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship sets R between entity sets A and B, the mapping cardinality must be one of the following:

- **One to One:** An entity in A is associated with at most one entity in B, an entity in B is associated with at most one entity in A. See Figure (a)
- **One to many:** An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A. See Figure (b)
- **Many to one:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A. See Figure (c)

- **Many to Many:** An entity in A is associated with any number of entities in B. An entity in B, can also be associated with any number of entities in A. See Figure (d)



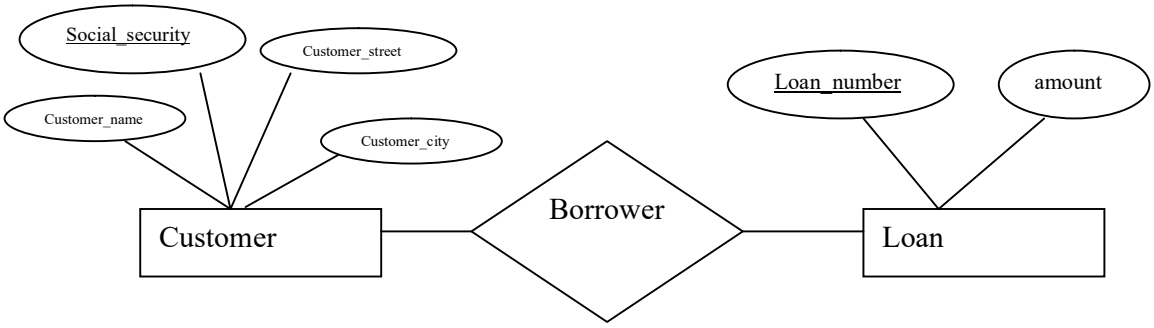
1.7.4 Entity Relationship Diagrams: The overall logical structure of a database can be expressed graphically by an E-R Diagram. The relative simplicity and pictorial clarity of this diagramming technique may well account in large part for the wide spread use of the E-R Model. Such a diagram consists of the following components:

- **Rectangles**, which represents entity sets.
- **Ellipses**, which represents attributes.
- **Diamonds**, which represents relationship sets.
- **Lines**, which link attributes to entity sets and entity sets to relationship sets.
- **Double** ellipses, which represents multivalued attributes.
- **Dashed** ellipses, which denote derived attributes.
- **Double Lines**, Which indicate total participation of an entity in a relationship set.

To distinguish the mapping cardinalities of relationship set, we draw either directed line (→) or an undirected line(-) between the relationship set and the entity set.

Consider the following E-R Diagram, which consists of two entity sets, customer and loan, related through a binary relationship set borrower. The attributes associated with customer are customer_name, social-security, customer-street, and customer-city. The attributes associated with loan are loan-number and amount.

The relationship set borrower may be many to many, one to many, many to one or one to one.



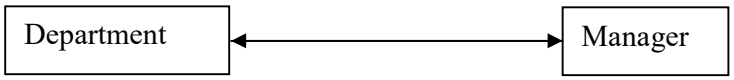
In the above E-R diagram, underlined attributes are acting as primary keys of the corresponding entity sets.

Direction : The direction of a relationship indicates the originating entity of a relationship . The entity from which a relationship originates is the parent entity ; the entity where the relationship terminates is the child entity .

The type of the relation is determined by the direction of line connecting relationship component and the entity . To distinguish different types of relation , we draw either a directed line or an undirected line between the relationship set and the entity set . Directed line is used to indicate one occurrence and undirected line is used to indicate many occurrences in a relation .

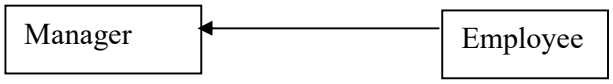
DEPARTMENT , MANAGER , EMPLOYEE , PROJECT

The relationship between a DEPARTMENT and a MANAGER is usually one-to-one ; there is only one manager per department . This relationship between entities is shown below. Each entity is represented by a rectangle and a direct line indicates the relationship between them . The relationship for MANAGER is both 1:1

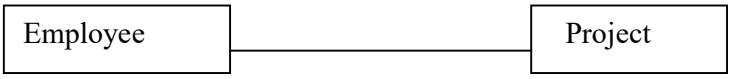


Note that a one to one relationship between two entity set does not imply that for an occurrence of an entity from one set at any time there must be an ocurrence of an entity in the other set. In the case of an organistion , there could be times when a department is without a manager or when an employee who is classified as a manager may be without a department.

A one to many relationship exists from the entity MANAGER to the entity EMPLOYEE because there are several employees reporting to the manager . As we have pointed out there could be an occurrence of the entity type MANAGER having zero occurrences of the entity type EMPLOYEE reporting to him or her . A reverse relationship , from EMPLOYEE to MANAGER would be many to one since a single manager may supervise many employees.



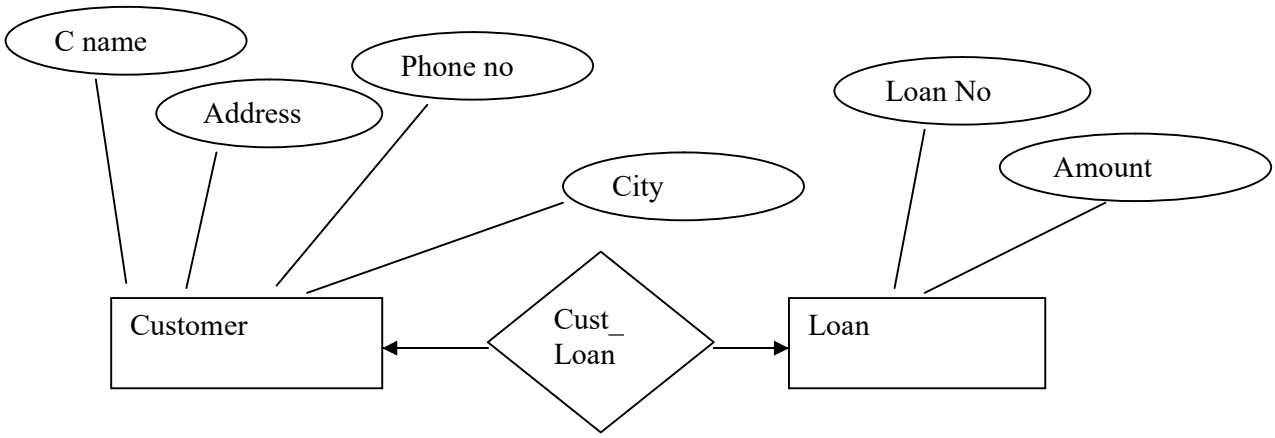
The relationship between the entity EMPLOYEE and the entity PROJECT can be derived as follows : Each employee could be involved in a number of different projects , and a number of employees could be working on a given project . This relationship between EMPLOYEE and PROJECT is many to many .It is illustrated as below .



These relations can also be expressed in terms of following examples .

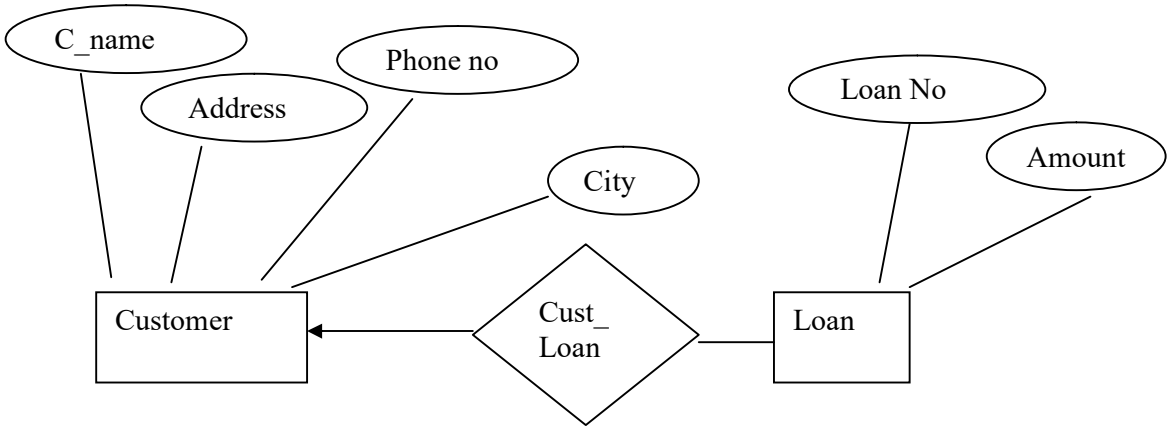
Consider a Customer – Loan relationship in which the loans are given to customers. Depending on the rules of the bank, one customer can take a single loan or multiple loans and relationship may be classified as

1:1 If one customer can take only one loan .



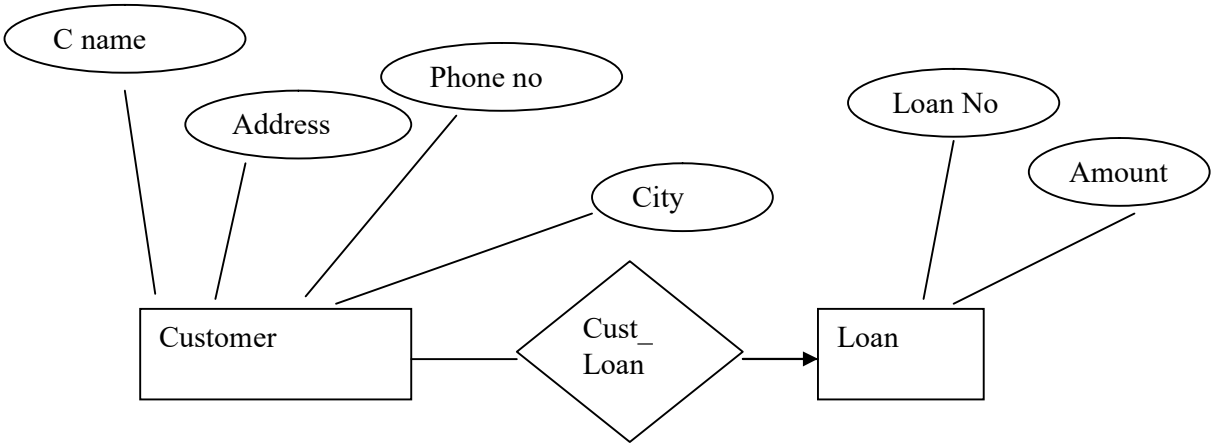
One-to-One Relationship

1:M If one customer can take multiple loans



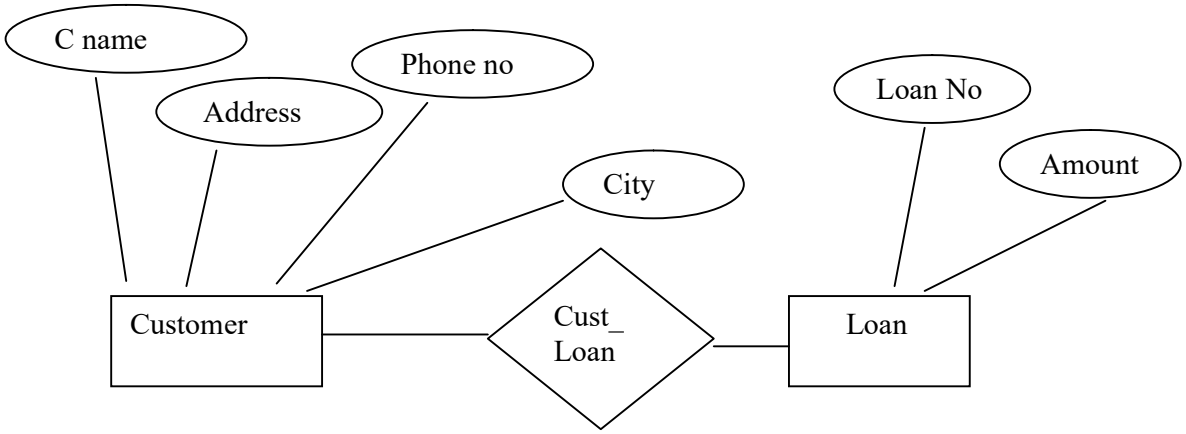
One-to-Many Relationship

M:1 If multiple customers participate for a single loan and one customer can take only one loan



Many-to-One Relationship

M:M If multiple customers participate for a single loan and one cutomer can take more than one loan .



Many-to-Many Relationship

1.7.5 Weak and Strong Entity Sets

An entity Set may not have sufficient attributes to form a primary key, such an entity set is termed as weak entity set. An entity set that has a primary key is termed as a strong entity set. For example, consider the entity payment, which has the three attributes: payment-number, payment-date, and payment-amount. Although each payment entity is distinct, payments for different loans may share the same payment number. Thus, this entity set does not have a primary key; it is a weak entity set. For a weak entity set to be meaningful, it must be part of a one-to-many relationship set. This relationship set should have no descriptive attributes, since any required attributes can

be associated with the weak entity set. A member of a strong entity set is a dominant entity, whereas a member of a weak entity set is a subordinate entity.

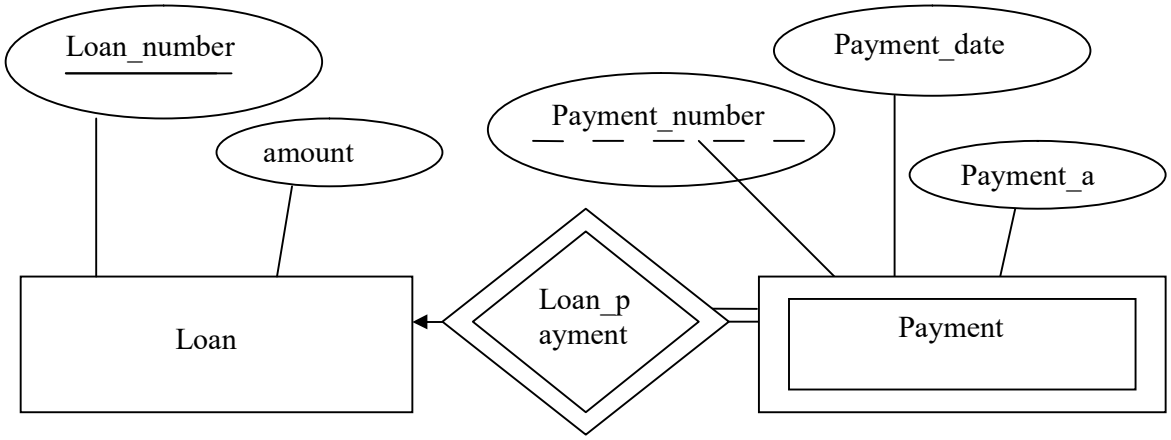
Although a weak entity set does not have a primary key, we nevertheless need a means of distinguishing among all those entities in the entity set that depend on one particular strong entity. The discriminator of a weak entity set is a set of attributes that allows this distinction to be made. For example, the discriminator of the weak entity set payment is the attribute payment number, since, for each loan, a payment number uniquely identifies one single payment for that loan. The discriminator of a weak entity set is also called the partial key of the entity set.

The Primary key of a weak entity set is formed from by the primary key of the strong entity set on which set is existence dependent, plus the weak entity set's discriminator. In the case of the entity set payment, its primary key is (loan-number, payment-number), where loan-number identifies the dominant entity of a payment, and payment-number distinguishes payment entities within the same loan.

The identifying dominant entity set is said to own the weak entity set that it identifies. The relationship that associates the weak entity set with an owner is the identifying relationship. In our example, loan-payment is the identifying relationship for payment.

A weak entity set in E-R Diagram is represented by a doubly outlined box, and the corresponding identifying relationship by a doubly outlined diamond.

The relationship between weak entity and strong entity set can be expressed with following example. In the following example loan-payment is the identifying relationship for payment entity. A weak entity set is represented by doubly outlined box corresponding identifying relation by a doubly outlined diamond. Here double line indicate total participation of weak entity in strong entity set it; means that every payment must be related via loan-payment to some account. The arrow from loan payment to loan indicates that each payment is for single loan. The discriminator of a weak entity set is underlined with dashed lines rather than solid line.

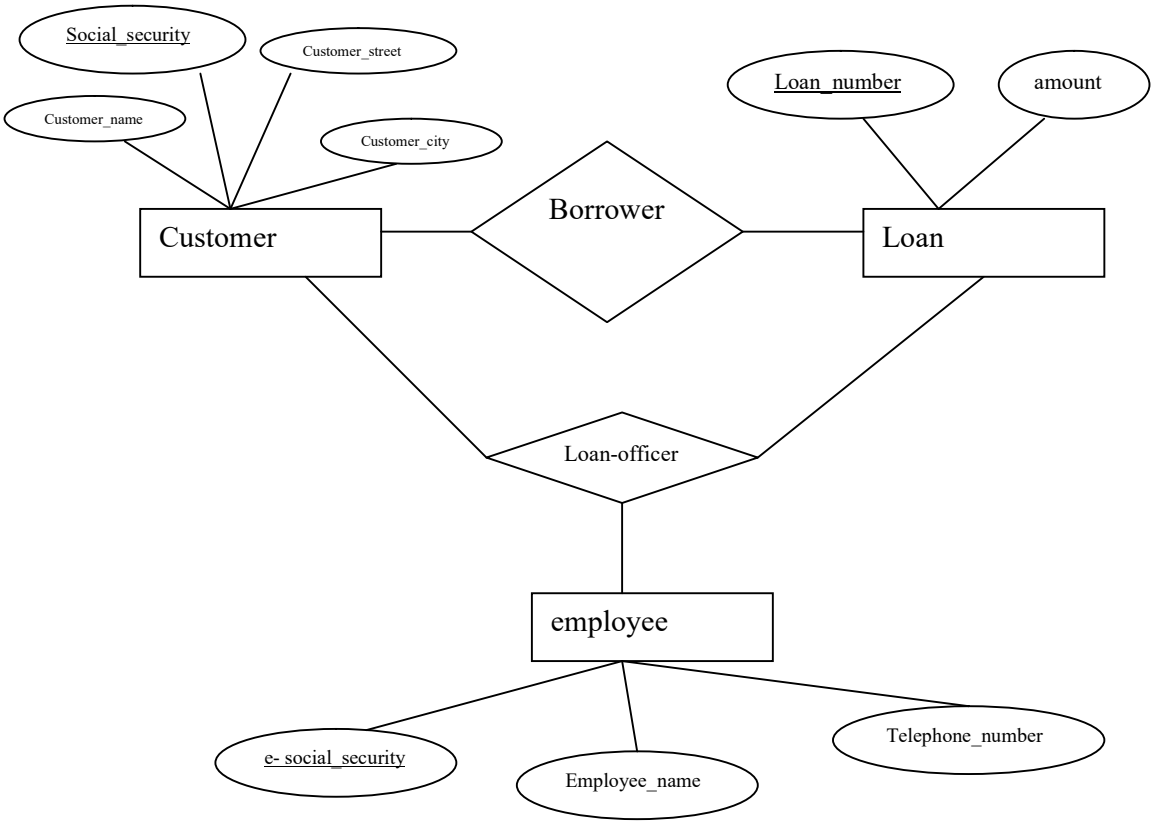


Relation between strong and weak entity set

1.7.6 Aggregation

One limitation of the E-R Model is that it is not possible to express relationships among relationships. To illustrate the need for such a construct, we consider a database describing information about customers and their loans. Suppose that each customer-loan pair may have a bank employee who is the loan officer for that particular pair. Using our basic E-R Diagram constructs, we obtain E-R diagram drawn in following figure (a) . It appears that the relationship sets borrower and loan-officer can be combined into one single relationship set. Nevertheless, we should not combine them, because doing so would obscure the logical structure of this schema. For example, if we consider the borrower and loan-officer relationship sets, then this combination specifies that a loan officer must be assigned to every customer-loan pair, which is not true. The separation into two different relationship sets solves this problem.

There is redundant information in the resultant figure, however, since every customer-loan pair in loan-officer is also in borrower. If the loan officer were a value rather than an employee entity, we could instead make loan-officer a multivalued attribute of the relationship borrower. But doing so makes it more difficult to find, for example, customer-laon pairs for which an employee is responsible. Since the loan officer is an employee entity, this alternative is ruled out in any case.The best way is to model a situation such as the one just described is to use aggregation. **Aggregation** is an abstraction through which relationships are treated as higher-level entities. Thus, for our example, we regard the relationship set borrower and the entity sets customer and loan as a higher-level entity set called borrower. Such an entity set is treated in the same manner as is any other entity set. A common situation for aggregation is shown in figure (b).



Figure(a): E-R diagram with redundant relationships

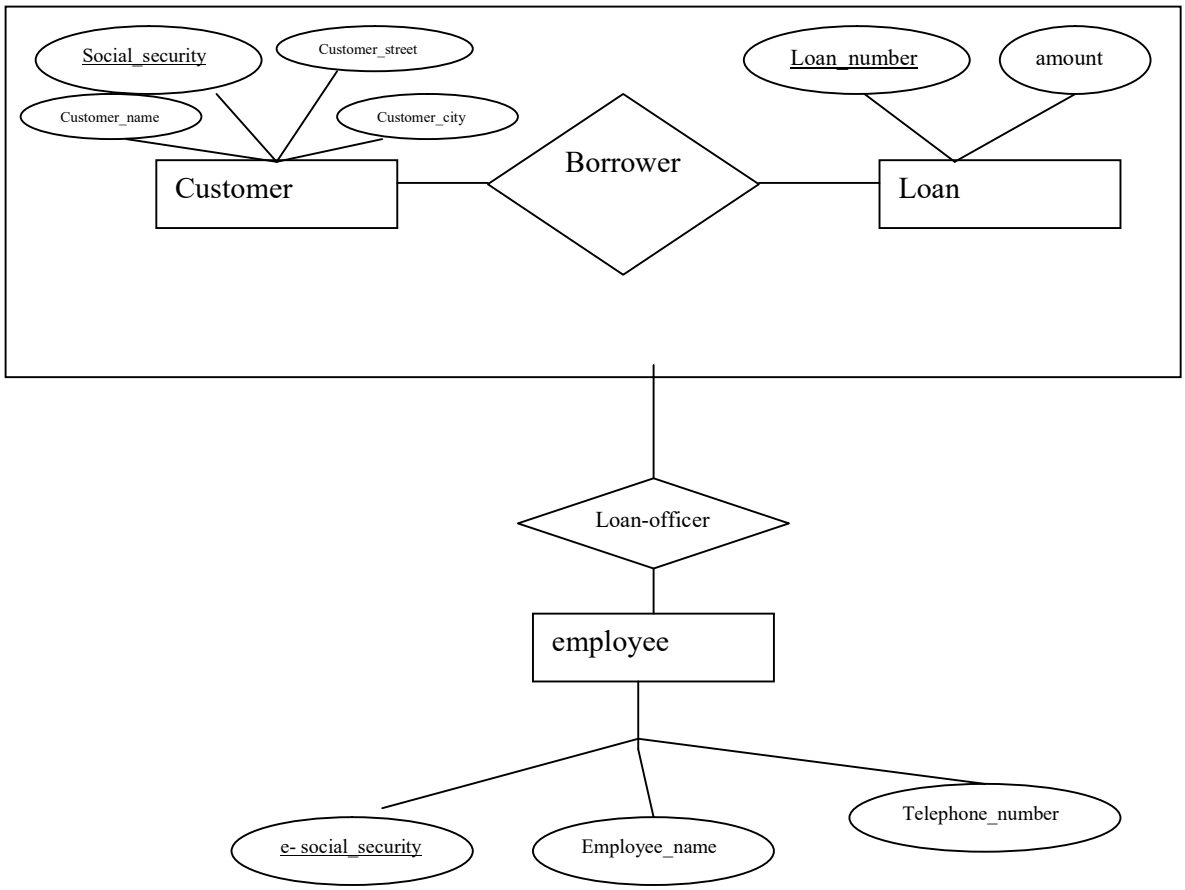
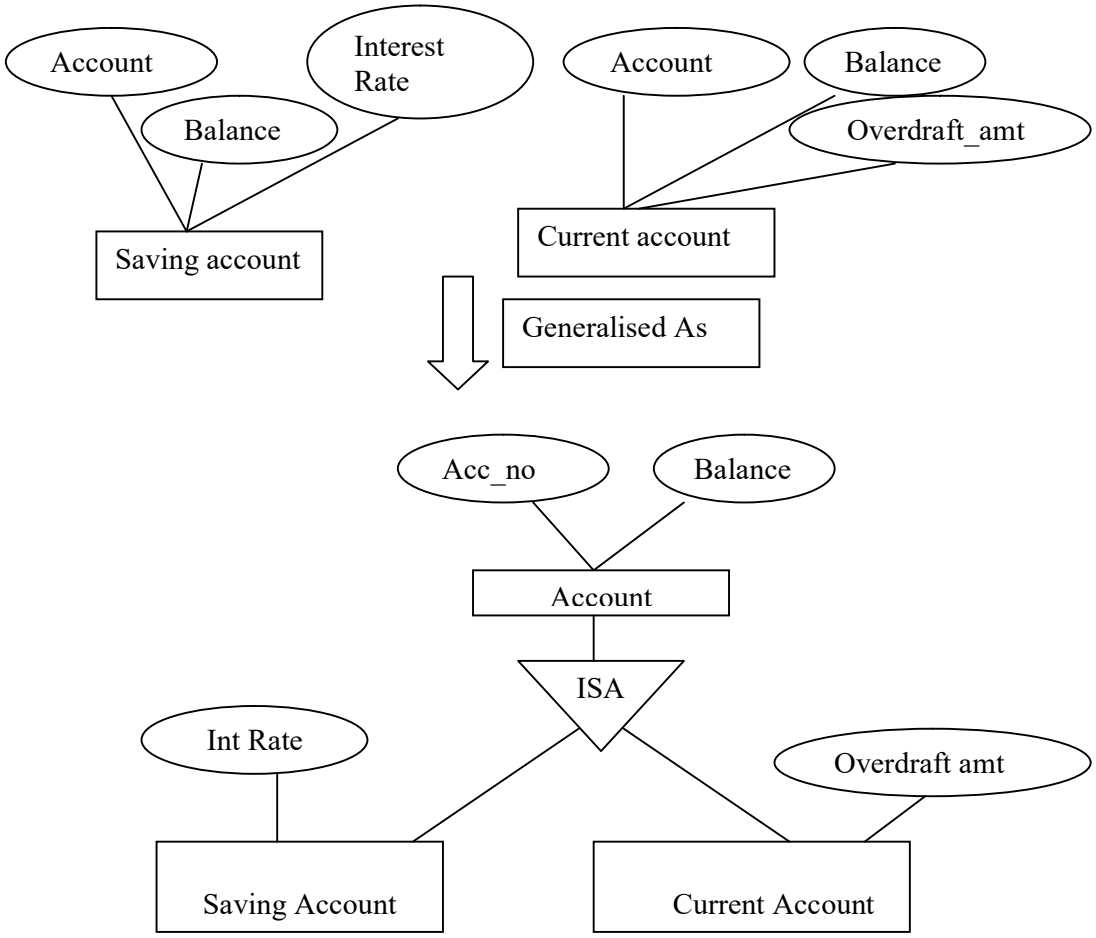


Figure (b) : E-R Diagram with aggregation

Generalization

A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into a higher level entity type called a supertype or generic entity . The lower level of entities become the subtype or categories to the supertype . Subtypes are dependent entities .

Generalization is used to emphasize the similarities among lower-level entity sets and to hide differences . It makes ER diagram simpler because shared attributes are not repeated . Generalisation is denoted through a triangle component labeled ‘ IS A ‘ , as shown below

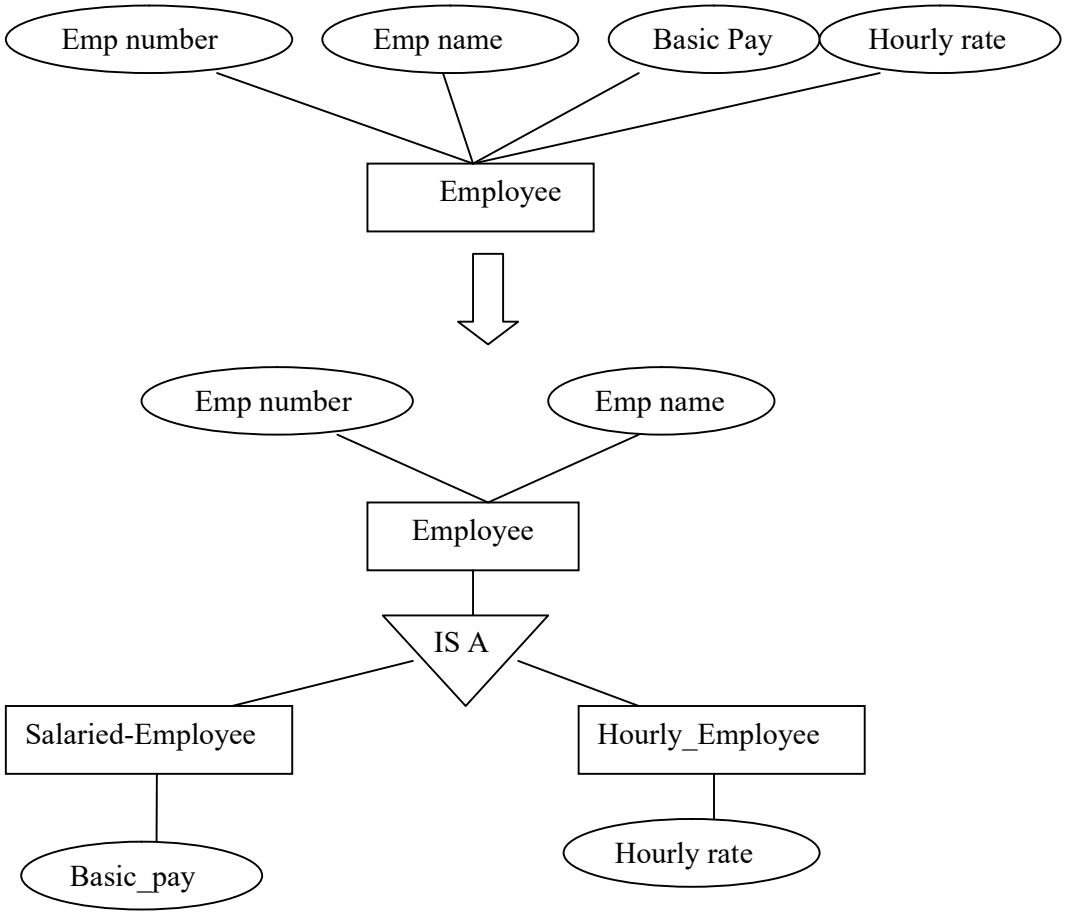


Account is the higher level entity set and Saving account and Current account are lower level entity sets .

Specialization

Specialization is the process of taking subsets of a higher level entity set to form lower level entity sets . It is a process of defining a set of subclasses of an entity type, which is called as superclass of the specialization . The process of defining subclass is based on the basis of some distinct characteristics of the entities in the superclass .

For example specialization of the Employee entity type may yield the set of subclasses namely Salaried_Employee and Hourly_Employee on the method of pay as shown below.



1.7.7 Summary

The entity-relationship model is based on the perception of a real world that consists of a set of basic objects called entities, and of relationships among these objects. There are three basic notions that the E-R data model employs – entity sets, relationship sets, and attributes. An entity is a thing or object in the real world that is distinguishable from all other objects. An entity set is a set of entities of the same type that share the same properties, or attributes. An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. A relationship is an association among several entities.

A relationship set is a set of relationships of the same type. Mapping Cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationship sets, although occasionally they contribute to the description of relationship sets that involve more than two entity sets. The overall logical structure of a database can be expressed graphically by an E-R Diagram. The relative simplicity and pictorial clarity of this

diagramming technique may well account in large part for the wide spread use of the E-R Model. An entity Set may not have sufficient attributes to form a primary key, such an entity set is termed as weak entity set. An entity set that has a primary key is termed as a strong entity set. One limitation of the E-R Model is that it is not possible to express relationships among relationships. This limitation can be removed through aggregation. Aggregation is an abstraction through which relationships are treated as higher-level entities.

1.7.8 Self Understanding:

1. Define entity, entity set, attribute, relationship and relationship set.
2. What do you mean by an attribute? Explain various types of attributes.
3. What is an E-R Diagram? Explain various constructs for drawing E-R Diagram.
4. Explain E-R Model.
5. Define Weak and strong entity sets with suitable examples.
6. Define aggregation with suitable example. Why it is needed?
7. Differentiate between weak and strong entity sets.
8. What do you mean by mapping cardinality? Explain various mapping cardinalities with suitable examples.

1.7.9 Further Readings:

Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.

C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.

Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

LESSON NO. 1.8 **AUTHOR : VISHAL GOYAL**

ENTITY RELATIONSHIP MODEL (PART-2)

Structure:

1.8.0 Introduction:

The refinement from an initial entity set into successive levels of entity subgroups represents a top-down design process in which, distinctions are made explicit. The design process may also be processed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set based on common features. Generalization is a simple inversion of Specialization. A database that conforms to an E-R database schema can be represented by a collection of tables. For each entity set and for each relationship set in the database, there is a unique table that is assigned the name of the corresponding entity set or relationship set. Each table has multiple columns, each of which has a unique name.

1.8.1 Objective

After completing this lesson, you will be able to:

- Explain the concept of generalization using suitable example.
- Explain the steps for converting E-R diagrams into tables.

1.8.2 Generalization

The refinement from an initial entity set into successive levels of entity subgroups represents a top-down design process in which, distinctions are made explicit. The design process may also be processed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set based on common features. The database designer may have first identified a checking-account entity set with the attributes account-number, balance, and overdraft-amount, and a savings-account entity set with the attributes account-number, balance, and interest-rate.

There are similarities between the checking-account entity set and the savings-account entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. Higher- and lower-level entity sets also may be designated by the terms superclass and subclass

respectively. The account entity set is the superclass of the savings-account and checking-account subclasses.

For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R Schema for an enterprise. In terms of the E-R diagram itself, we do not distinguish between specialization and generalization. New levels of entity representation will be distinguished (specialization) or synthesized (generalization) as the design schema comes to express fully the database application and the user requirements of the database. Differences in the two approaches may be characterized by their starting point and overall goal.

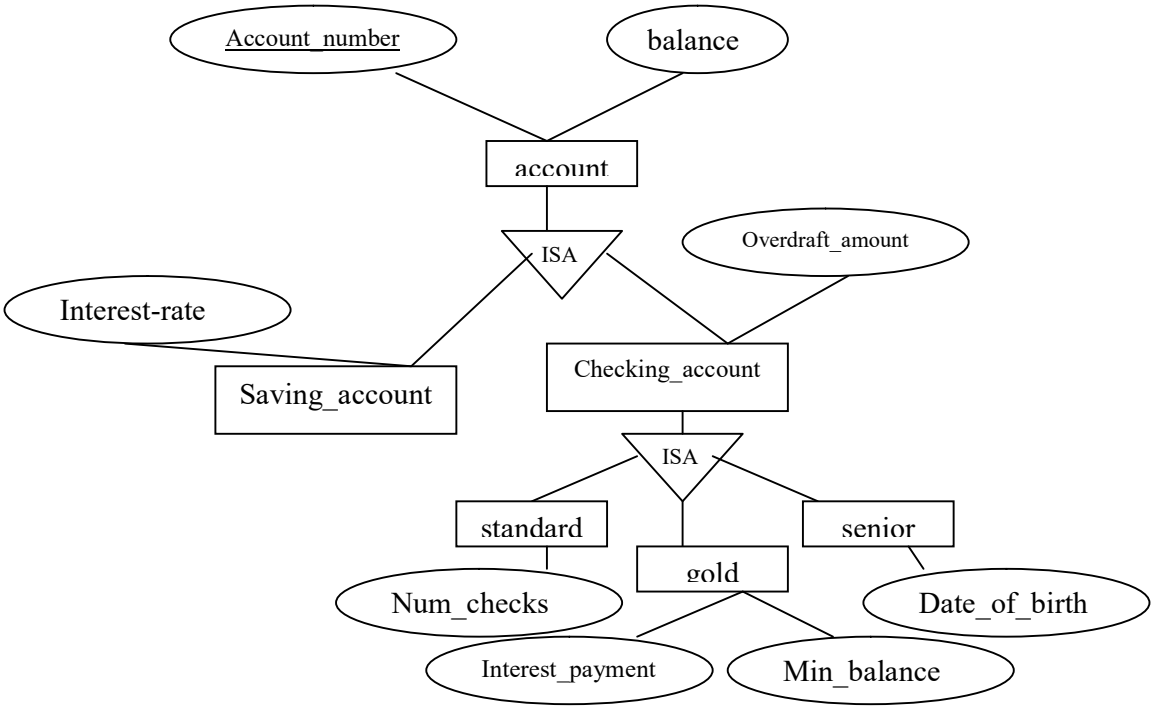
Specialization stems from a single entity set, it emphasizes differences among entities within the set by creating distinct lower-level entity sets. These lower-level entity sets may have attributes, or may participate in relationships, that do not apply to all the entities in the higher-level entity set. Indeed, the reason a designer applies specialization is to represent such distinct features. If savings-account and checking-account did not each have unique attributes, there would be no need to specialize the account entity set.

Generalization proceeds from the recognition that a number of entity sets share some common features. Based on their commonalities, generalization synthesizes these entity sets into a simple, higher-level entity set. Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences; it also permits an economy of representation in that shared attributes are not repeated.

A crucial property of the higher-level and lower-level entities created by specialization and generalization is attribute inheritance. The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets. For example: savings-account and checking-account inherit the attributes of account. Thus saving-account is described by its account-number, balance and Interest rate and Checking account is described by its account-number, balance, and overdraft-amount attributes. A lower-level entity set also inherits participation in the relationship sets in which its higher-level participates. Both the savings-account and checking-account entity sets participate in the depositor relationship set. Attributes inheritance applies through all tiers of the lower-level entity sets. The standard, gold, and senior lower-level entity sets inherit the attributes and relationship participation of both checking-account and account.

Whether a given portion of an E-R model was arrived at by specialization or generalization, the outcome is basically the same:

- A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets.
- Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set.



1.8.3 Converting E-R diagrams into tables

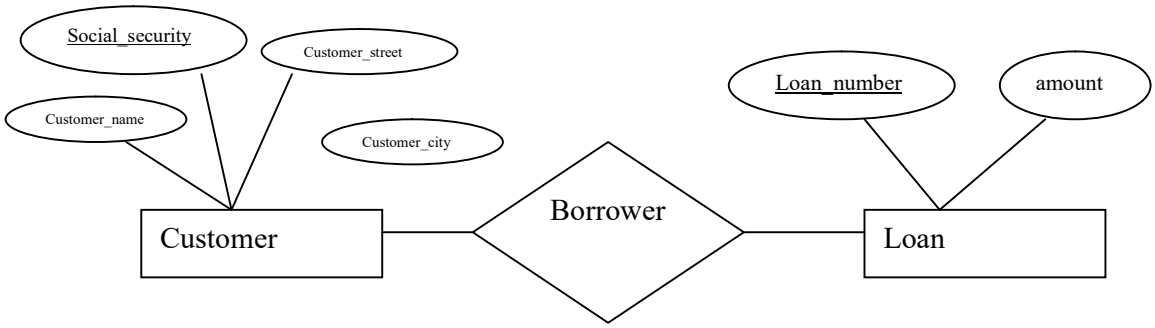
A database that conforms to an E-R database schema can be represented by a collection of tables. For each entity set and for each relationship set in the database, there is a unique table that is assigned the name of the corresponding entity set or relationship set. Each table has multiple columns, each of which has a unique name.

Both the E-R model and the relational database are abstract, logical representations of real world enterprises. Because the two models employ similar design principles, we can convert an E-R design into a relational design. Consequently a database representation from an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram. In the following section, we will explain how to convert the E-R diagram or schema into tables.

1.8.3.1 Tabular representation of Strong Entity Sets

Let E be a strong entity set with descriptive attributes a_1, a_2, $a3$.....,a_n. We represent this entity by a table called E with n distinct columns, each of which corresponds to one of the attributes of E. Each row in this table corresponds to one entity of the entity set E.

Consider the entity set loan of the E-R diagram shown below:



This entity set has two attributes: loan_number and amount. We represent this entity set by table loan, with two columns, as shown in following figure. The row (L-17,1000) in the loan table means that loan number L-17 has a amount of Rs. 1000. We can add a new entity to the database by inserting a row into a table. We can also delete or modify rows.

Table : loan	
Loan_number	**amount**
L-17	1000

Let D_1 denote the set of all loan numbers and let D_2 denote the set of all balances. Any row of the loan table must consist of a 2-tuple (v_1,v_2) where v_1 is a loan, v_1 belongs to D_1 and v_2 is an amount where v_2 belongs to D_2. In general, the loan table will contain only a subset of the set of all possible rows. We refer to the set of all possible rows of loan as the Cartesian product of D_1 and D_2 denoted by $D_1 \text{ X } D_2$.

In general, if we have a table of n columns, we denote the Cartesian product of D_1, D_2, D_3, … D_n by $D_1 \text{X } D_2 \text{ X } D_3$ …..X D_n.

Similarly, another example, consider the entity set customer of the E-R Diagram shown above. This entity set has the attributes customer_name, social_security, customer_street, customer_city. The table corresponding to customer has four columns as shown in Figure below:

Table : Customer			
Customer_name	**Social_security**	**Customer_street**	**Customer_city**
Suresh	123-4-567	Phase I	Patiala

1.8.3.2 Tabular Representation of Weak entity Sets

Let A be a weak entity set with attributes a_1, a_2, $a3$……,a_n. Let B be the strong entity set on which A is dependent. Let the primary key of B consists of attributes b_1, b_2, $b3$……,b_n.. We represent the entity set A by a table called A with one column for each attribute of the set:

$$\{ a_1, a_2, a3\text{……},a_n\} \text{ U } \{ b_1, b_2, b3\text{……},b_n \}$$

As an illustration, consider the entity set payment consisting of attributes payment_number, payment_date, payment_amount. The primary key of the loan entity set, on which payment is dependent, is loan_number. Thus , payment is represented by a table with four columns labeled loan_number, payment_number, payment_date, payment_amount as shown in following figure:

Table: payment			
Loan_number	**Payment_number**	**Payment_date**	**Payment_amount**
L-17	67	10-11-2006	1000

1.8.3.2 Tabular Representation of Relationship sets

Let R be a relationship set, let a_1, a_2, a3.....,a_n be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R, and let descriptive attributes (if any) of r be b_1, b_2, b3.....,b_n.. We represent this relationship set by a table called R with one column for each attribute of the set:

$$\{ a_1, a_2, a3.....,a_n\} \cup \{ b_1, b_2, b3.....,b_n \}$$

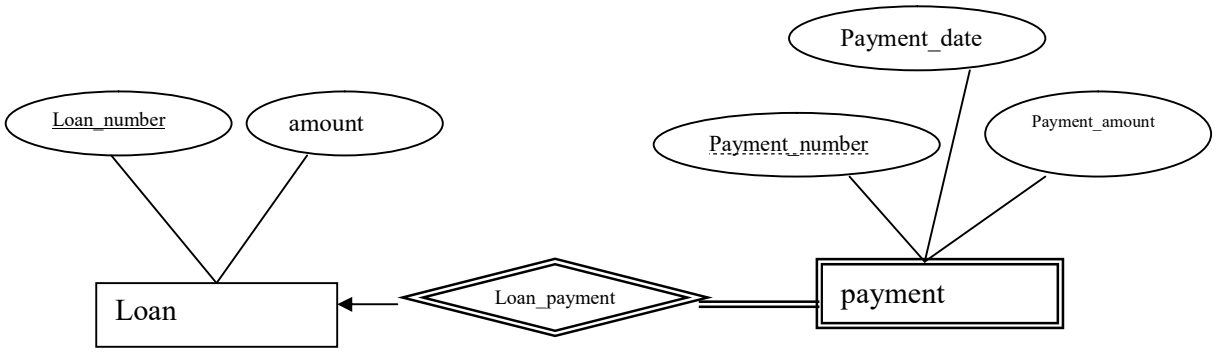
As an illustration, consider the relationship set borrower in the E-R Diagram shown above. This relationship set involves the following two entity sets:

- Customer, with the primary key social security
- Loan, with the primary key loan_number

Since the relationship set has no attributes, the borrower table has two columns labeled social_security and loan_number as shown in figure below:

Table : borrower	
Social_security	**Loan_number**
123-4-567	67

The case of a relationship set linking a weak entity set to the corresponding strong entity set is special. As we noted earlier, these relationships are many-to-one and have no descriptive attributes. Furthermore, the primary key of a weak entity set includes the primary key of the strong entity sets. In the E-R diagram shown below, the weak entity set payment is dependent on the strong entity set loan via the relationship set loan-payment. The primary key of payment is {loan_number, payment_number} and the primary key of loan is {loan_number}. Since loan_payment has no descriptive attributes, the table for loan_payment would have two columns, loan_number and payment_number. The table for the entity set payment has four columns, loan _number, payment_number,payment_date,payment_amount. Thus, the loan-payment table is redundant. In general, the table for the relationship set linking a weak entity set with its corresponding strong entity set is redundant and does not need to be present in a tabular representation of E-R diagram.



Multivalued Attributes

We have seen that attributes in an E-R diagram generally map directly into columns for the appropriate tables. Multivalued attributes, however, are an exception; new tables are created for these attributes.

For a multivalued attribute M, we create a table T with a column C that corresponds to M and columns corresponding to primary key of the entity set or relationship set of which M is an attribute. For example, consider the E-R diagram that includes the multivalued attribute dependent_name. For this multivalued attribute, we create a table dependent_name, with columns dname, referring to the dependent_name attribute of employee, and e_social_security, representing the primary key of the entity set employee. Each dependent of an employee is represented as a unique row in the table.

1.8.3.4 Tabular Representation of Generalization

There are two different methods for transforming to a tabular form an E-R diagram that includes generalization.

1. Create a table for the higher-level entity set. For each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the primary key of the higher-level entity set. Thus, for the E-R diagram in section 1.8.2, we have three tables:
   - Account, with attribute account_number and balance
   - Savings_account, with attributes account_number and interest_rate
   - Checking_account, with attributes account_number and overdraft_amount
2. If the generalization is disjoint and complete – that is if no entity is a member of two lower-level entity sets directly below a higher-level entity set, and if every entity in the higher-level entity set is also a member of one of the lower-level entity sets – then an alternative representation is possible. Here, create no table for the higher level entity set. Instead, for each lower-level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the higher level entity set. Then, for the E-R diagram in section 8.2 we have two tables:

- saving_account, with attributes account_number, balance, and interest_rate.
- Checking_account, with attributes accounts_number, balance, and overdraft_amount

The saving_account and checking_account relations corresponding to these tables both have account-numbers as the primary key.

If the second method were used for an overlapping generalization, some values such as balance would be stored twice unnecessarily. Similarly, If the generalization were not complete—that is, if accounts were neither savings nor checking accounts—then such accounts could not be represented with the second option.

1.8.4 Summary

Specialization stems from a single entity set, it emphasizes differences among entities within the set by creating distinct lower-level entity sets. Generalization proceeds from the recognition that a number of entity sets share some common features. Based on their commonalities, generalization synthesizes these entity sets into a simple, higher-level entity set. Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences; it also permits an economy of representation in that shared attributes are not repeated. Generalization is a simple inversion of Specialization. Both the E-R model and the relational database are abstract, logical representations of real world enterprises. Because the two models employ similar design principles, we can convert an E-R design into a relational design. Consequently a database representation from an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

1.8.5 Self Understanding

1. Explain the concept of generalization with suitable example.
2. Explain the steps for converting E-R Diagrams into tables.

1.8.6 Further Readings

Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.

C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.

Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

LESSON NO. 1.9 **AUTHOR : VISHAL GOYAL**

OVERVIEW OF NETWORK AND HIERARCHIAL DATA MODEL

Structure:

1.9.0 Introduction:

The network data model represents entities by records and expresses relationships between entities by means of sets implemented by the use of pointers or links. The model allows the representation of an arbitrary relationship. The DBTG proposal places a number of restrictions on the use of the links. A set is a means of representing a one-to-many relationship between record types. A set type can have an arbitrary number of occurrences.

Like the network data model, the hierarchical data model uses the records and pointers or links to represent entities and relationships among them. However, unlike the network data model, the data structure used is a rooted tree with a strict parent-to-child ordering. IBM's IMS DBMS includes the features of Hierarchical Model

1.9.1 Objective

After completing this lesson, you will be able to:

- Explain the basics of Network and Hierarchical Model
- Understand the implementation of Network and Hierarchical Model

1.9.2 The Network Data Model

The network data model was formalized in the late 1960s by the Database Task Group of the Conference on Data System Language (DBTG / CODASYL). Their first report which has been revised a number of times, contained detailed specifications for the network data model. The specifications contained in the report and its subsequent revisions have been built on commercial DBMS using DBTG model.

The Network data Model (NDM) represents data for any entity set by a logical record type. The data for an instance of an entity set is represented by a record occurrence of the record type.

DBTG Model:

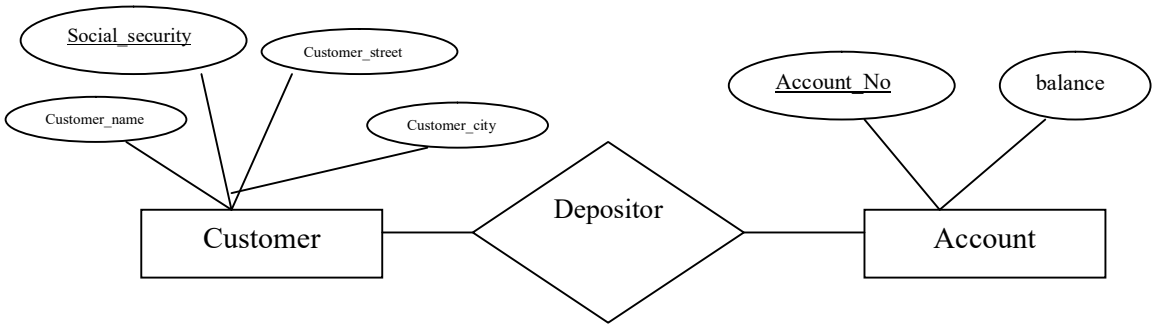
The DBTG model uses two different structures to represent the database entities and relationships between the entities, namely record type and set type.

A record type is used to represent an entity type. It is made up of a number of data items that represent the attributes of the entity.

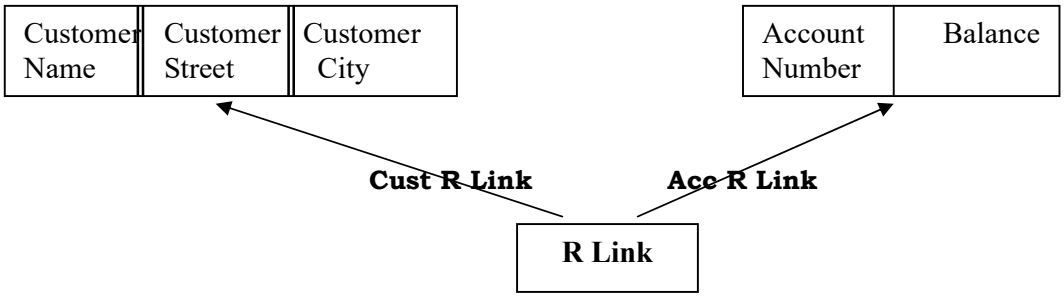
A set type is used to represent a directed relationship between two record types, the so called owner record type, and the member record type. The set type, like the record type, is named and specifies that there is one-to-many relationship (1: M) between the owner and member record types. The set type can have more than one record type as its member, but only one record type is allowed to be the *owner in a given set type*. A database could have one or more occurrences of each of its record and set types. *An occurrence of a set type* consists of an occurrence of the owner record type and any number of occurrences of each of its member record types. A record type cannot be a member of two distinct occurrences of the same set type.

In this model, only many-to-one links can be used. One-to-one links are represented as many-to-one links. Many-to-many links are disallowed to simplify the implementation.

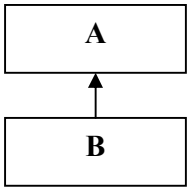
For example:



In the above E-R Diagram, depositor relationship is many-to-many and to transfer the relationship into DBTG model, we have to convert this many-to-many relationship into many-to-one relationships as only many-to-one relationships are allowed in the DBTG model. Thus, we must create a new dummy record type, R link that may either have no fields or have a single field containing an externally defined unique identifier, and two many-to-one links – cust R Link and Acc R Link as follows:



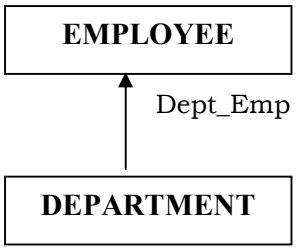
In the DBTG model, a data structure diagram consists of two record types and are linked to each other through many-to-one relationships. The general form of this structure is



This structure is referred to as DBTG set.

DBTG Set:

Bachman introduced a graphical means called a data structure diagram to denote the logical relationship implied by a set. Here a labeled rectangle represents the corresponding entity or record type. An arrow that connects two labeled rectangles represents the corresponding entity or record type. The arrow direction is from the owner record type to the member record type. Figure shows two record types (DEPARTMENT and EMPLOYEE) and the set type DEPT_EMP, with DEPARTMENT as the owner record type and EMPLOYEE as the member record type.



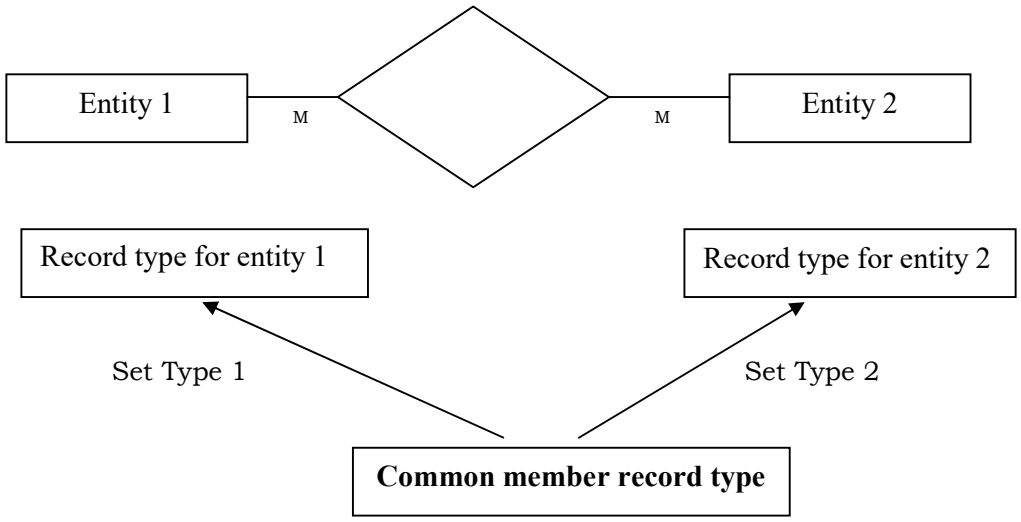
The data structure diagrams have been extended to include field names in the record type rectangle, and the arrow is used to clearly identify the data fields involved in the set association. A one-to-many (1:M) relationship is shown by a set type arrow that starts from the owner field in the owner record type. The arrow points to the member field within the member record type. The fields that support the relationship are clearly identified.

Each entity type in E-R Diagram is represented by a logical record with the same name. The attributes of the entity are represented by data fields of the record. We use the term *logical record* to indicate that the actual implementation may be quite different.

The conversion of the E-R Diagram into a Network database consists of converting each 1:M binary relationship into a set. If there is a 1:M binary relationship R_1 from entity type E_1 to the entity type E_2, then the binary relationship is represented by a set. An instance of this would be S_1 with an instance of the record type corresponding to the entity E_1 as the owner and one or more instances of the record type corresponding to entity E_2 as the member. If a relationship has attributes, unless the attributes can be assigned to the member record type, they have to be maintained in a separate logical record type created for this purpose. The introduction of this

additional record type requires that the original set be converted into two symmetrical sets, with the record corresponding to the attributes of the relationship as the member in both the sets and the records corresponding to the entities as the owners.

Each many-to-many relationship is handled by introducing a new record type to represent the relationship wherein the attributes, if any of the relationship are stored. We then create two symmetrical 1: M sets with the member in each of the sets being the newly introduced record type. The conversion of a many-to-many relationship into two one-to-many sets using a common member record type as shown in figure below:



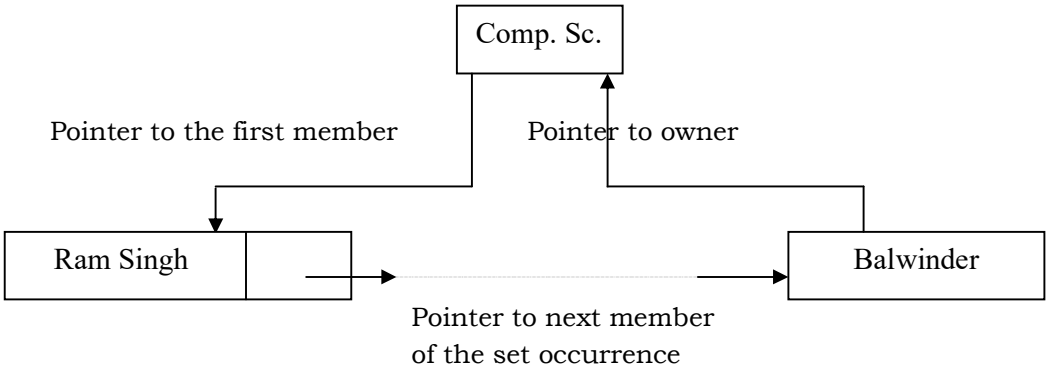
In the network model, the relationships as well as the navigation through the database are predefined at database creation time.

Implementation of the Network Data Model:

The record is basic unit to represent data in the DBTG network database model. The implementation of the one-to-many relationships of a set is represented by linking the members of a given occurrence of a set to the owner record occurrence. The actual method of linking the members of a given record occurrence to the owner is immaterial to the user of the database; however, we can assume that the set is implemented using a linked list. The list starts at the owner record occurrence and links all the member record occurrences with the pointer in the last record occurrence leading back to the owner record.

Figure below shows the implementation of the set occurrence DEPT-EMP where the owner record is Comp.Sc. And the member records are the instances Ram Singh and Balwinder. Note that for simplicity we have shown only one of the record fields of each record. This method of implementation assigns one pointer (link) in each record for each set type in which the record participates and, therefore allows a record occurrence to participate in only one occurrence of a given set type. Any other method of implementing

the set construct in a database management system based on the DBTG proposal is, in effect, equivalent to the linked list method.



A second form of network implementation, especially useful for M:M relationships, is a bit map, which is depicted in figure. A bit map is a matrix created for each relationship. Each row corresponds to the relative record number of a target record of a relationship. A 1 bit in a cell for row X and the column Y means that records corresponding to row X and column Y are associated in this relationship; a zero means no association.

	1	2	.....	Y	.....
1	1	0	.....	0	.....
2	1	0	.....	0	.....
.	.	.	.	.	.
.					
.					
X	1	0	.....	1	.....
.					
.					

For example, figure below indicates that PRODUCT with relative record number X is related to VENDOR with relative record numbers 1 and Y. Bit maps are powerful data structures for the following reasons:

1. Any record type(s) can be included in rows and columns.
2. 1:1, 1:M, M:1 relationship can all be represented.
3. Rows and columns can be logically manipulated by Boolean operators to determine records that satisfy complex associations.
4. A bit map can be manipulated equally as well in either a row or column access and can be easily extended for n-ary relationships.

Advantages and Disadvantages of Network model

The main advantages of network model are :

Simplicity

One of the basic advantages of network model is its simplicity . Network model is conceptually very simple to design

Capability to handle more relationship types :

The network model can handle the one-to-many (1 : M) and many to many relationships, which is a real help in modeling the real life situations.

Data Integrity

Integrity actually means that the data stored in the database is consistent. It actually ensures the accuracy and efficiency of data stored. Data integrity is also one of the advantages of network model. Network model does not allow member to exist without an owner. Thus a user must first define the owner record and then the member record, this ensures data integrity.

Data independence

Network model allows isolating the programs from complex physical storage details .

Database Standards

One of the major drawbacks of other models were the non-availability of universal standards for database design and modeling. The network model is based on the standards formulated by the DBTG and augmented by ANSI/SPARC (American National Standards Institute/ Standards Planning and Requirements Committee) in the 1970s. All the network database management systems conformed to these standards. These standards included a Data Definition Language (DDL) and Data Manipulation Language (DML), thus greatly enhancing database administration and portability.

Disadvantages of Network model

Even though network database model was significantly better than the early models it also had many drawbacks . Some of them are

System complexity:

All the records are maintained using pointers and hence the whole database structure becomes very complex.

Operational Anomalies

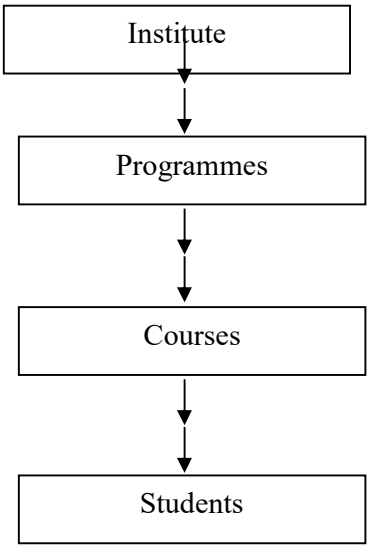
In network model insertion, deletion, and updating operations of any record require large number of pointer adjustments, which makes the implementation very complex and complicated .

Absence of structural independence

Since the data access method in the network database model is a navigational system, making structural changes to the database is very difficult in most cases and impossible in some cases. If changes are made to the database structure then all the application programs need to be modified before they can access data. Thus, even though the network database model succeeds in achieving data independence, it still fails to achieve structural independence.

1.9.3 THE HIERARCHICAL DATA MODEL

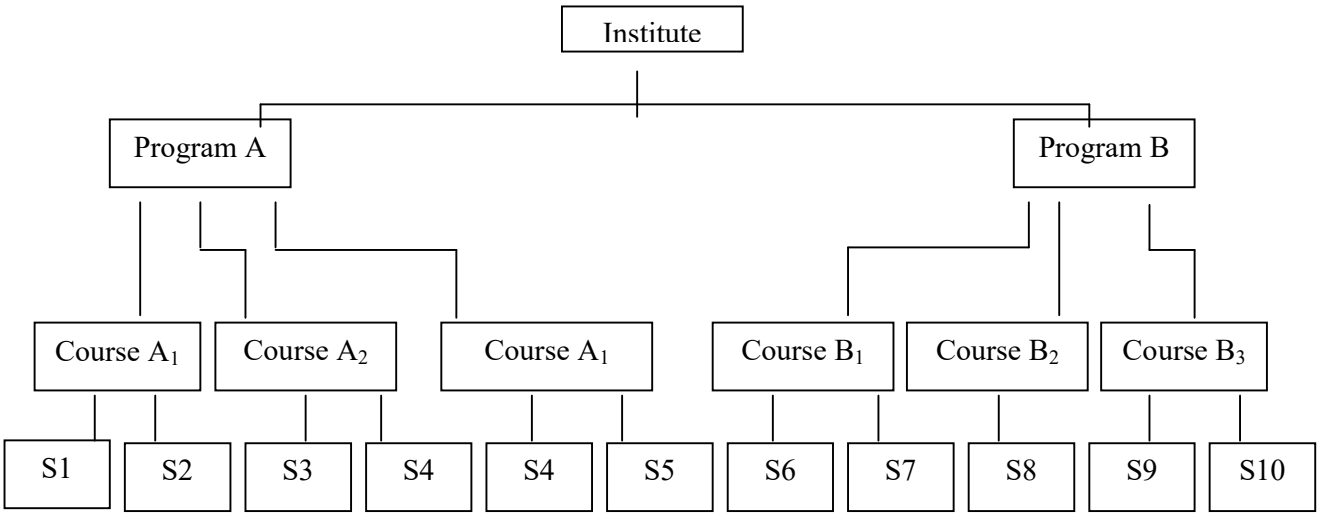
A DBMS belonging to the hierarchical data model uses tree structures to represent relationship among records. Tree structures occur naturally in many data organizations because some entities have an institute hierarchical order. For example, an institute has a number of programmes to offer. Each program has a number of courses. Each course has a number of students registered in it. The following figure depicts the four entity types Institute, Programmes, Courses and Students to make up the four different levels of hierarchical structure. The figure below shows an example of database occurrence for an institute. A database is a collection of database occurrence.



Definition: A hierarchical database therefore consists of a collection of records which are connected with each other through links. Each record is a collection of fields (attributes), each of which contains one data value. A link is an association between precisely two records.

A tree structure diagram serves the same purpose as an entity-relationship diagram; namely it specifies the overall logical structure of the database.

The following figure shows typical database occurrence of hierarchical structure (tree structure).



The hierarchical data model has the following features:

- Each hierarchical tree can have only one root record type and this record type does not have a parent record type.
- The root can have any number of child record types and each of which can itself be a root of a hierarchical sub tree.
- Each child record type can have only one parent record type; thus a M:M relationship cannot be directly expressed between two record types.
- Data in a parent record applies to all its children records.
- Each record occurrence must have a parent record occurrence; deleting a parent record occurrence requires deleting its entire children record occurrence.

Replication Vs Virtual Record

The hierarchical model, like the network model cannot support a many-to many relationship directly. In the network model the many-to many relationship is implemented by introducing an intermediate record and two one-to-many relationship. In the hierarchical model, the many-to many relationship can be expressed using one of the following methods: replication or virtual record. When more than one employee works in a given department, then for the hierarchical tree with EMPLOYEE as the root node we have to replicate the record for the department and have this replicated record attached as a child to the corresponding occurrence of the EMPLOYEE record type.

Replication of data would mean a waste of storage space and could lead to data inconsistencies when some copies of replicated data are not updated. The other method for representing the many-to-many relationship in the hierarchical data model is to use an indirect scheme similar to the network approach. In the hierarchical model the solution is to use the so-called virtual record. A virtual record is essentially a record containing a pointer to an occurrence of an actual physical record type. This physical record type is called the logical parent and the

virtual record is the logical child. When a record is to be replicated in several database trees, we keep a single copy of that record in one of the trees and replace, each other record with a virtual record containing a pointer to that physical record. To be more specified, lit R be a record type that is replicated in several hierarchical diagrams say H_1, H_2….H_n. To eliminate replication we create a new virtual record type

Virtual record type virtual – R. Virtual –will contain no data.

The Accessing of Data Records in Hierarchical Data Structure

The tree type data structures used to represent hierarchical data model shows the relationships among the parents, children, cousins, uncles, aunts, and siblings. A tree is thus a collection of nodes. One node is designated as the root node; the remaining nodes form trees or subtrees.

An ordered tree is a tree in which the relative order of the subtrees is significant. This relative order t only signifies the vertical placement or level of the subtrees but also the left to right ordering. Figures (a) and (b) below give two examples of ordered trees with A as the root node and B,C, and D, in turn, are root nodes or subtree with children nodes (E,F), (G)and (H,I,J) respectively. The significance in the ordering of the subtrees in these diagrams is discussed below.

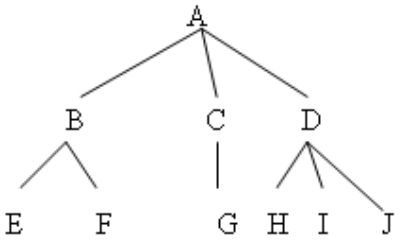


Figure (a)

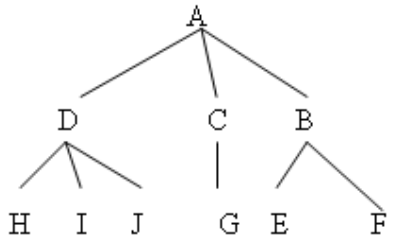


Figure (b)

Traversing an ordered tree can be done in a number of ways. The order of processing the nodes of the tree depends on whether or not one processes the node before the node's subtree and the order of processing the subtrees (left to right or right to left). The usual practice is the so-called preorder traversal in which the node is processed first, followed by the leftmost subtree not yet processed.

The preorder processing of the ordered tree of figure (a) below will process the nodes in the sequence A,B,E.F,C,G,D,H,I,J.

The significance of the ordered tree becomes evident when we consider the sequence in which the nodes could be reached when using a given tree traversing strategy. For instance, the order in which the nodes of the hierarchical tree of figure (b) are processed using the preorder processing strategy is not the same as the order for figure (a) above , even though the tree of part b contains the same nodes as the tree of part a.

Two distinct methods can be used to implement the preorder sequence in the ordered tree. The first method, shown in figure (c) below uses hierarchical pointers to implement the ordered

tree of figure (a) above .Here the pointer in each record points to the next record in the preorder sequence.

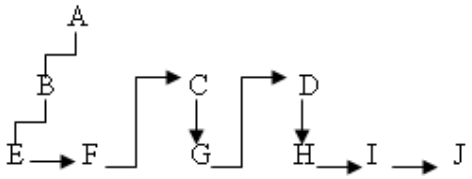


Figure (c): Preorder Traversal

The second method, shown in figure (d) uses two types of pointers; the child and the sibling pointers.

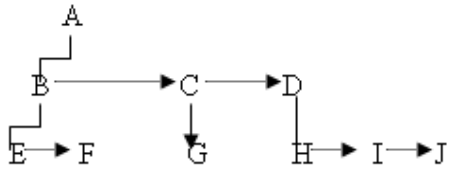


Figure (d): Child/ Sibling Pointers

The child pointer is used to point to the leftmost child and the sibling pointer is used to point to the right sibling. The siblings are nodes that have the same parent. Thus, the binary tree corresponding to tree in the figure (a) above is obtained by connecting together all siblings of a node and deleting all links from a node to its children except for the link to its leftmost child, using this transformation, we obtain the tree representation as shown in figure (d).

Implementation of the Hierarchical Data Model

Each occurrence of a hierarchical tree can be stored as a variable length physical record, the nodes of the hierarchy being stored in preorder. In addition the stored record contains a prefix field. This field contains control information including pointers, flags, locks and counters, which are used by DBMS to allow concurrent usage and enforce data integrity.

A number of methods could be used to store the hierarchical trees in the physical medium. It affects not only the performance of the system but also the operations that can be performed on the database, For example, if each occurrence of the hierarchical tree is stored as a variable length record on magnetic tape like device, the DBMS will allow only sequential retrieval and insertion or modification may be disallowed or performed lane by recreating the entire database. The insertion and modification storage of the hierarchical database on a direct access device allows an index structure to be supported for the root nodes and allows direct access to an occurrence of the hierarchical definition tree of figure (a) using the variable length record approach is given in the following figure:

$\{a_1\ b_1\ e_{11}\ e_{12}\ f_1\ a_1\ b_2\ e_{21}\ e_{22}\ f_2\ c_1\ g_1\ d_1\ h_1\ I_1\ J_1\ d_2\ h_{21}\ h_{22}\ I_2\ J_2\}$

Figure (e): Sequential storage of hierarchical database

The hierarchy can also be represented using pointer of either preorder hierarchical type or child/sibling type. In the hierarchical type of pointer, each record occurrence has a pointer that points to the next record in the pre order sequence. The child pointer points to its leftmost child record occurrence. The sibling pointer points to its right sibling (or twin). A record has one sibling pointer and its many child pointers as the number of child types associated with the node corresponding to the record. The following two figures (f) and (g) illustrate preorder hierarchical pointer and child sibling pointers respectively of hierarchical tree shown in figure (e):

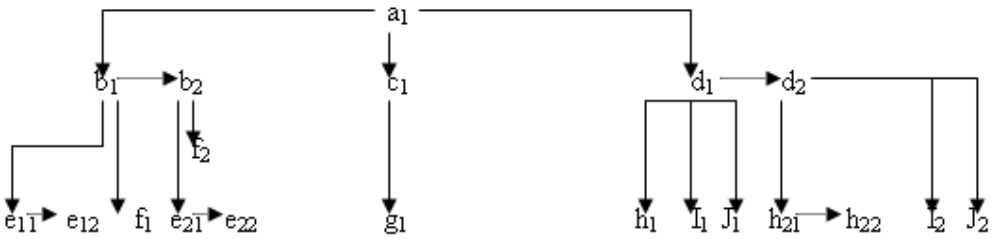


Figure (f)

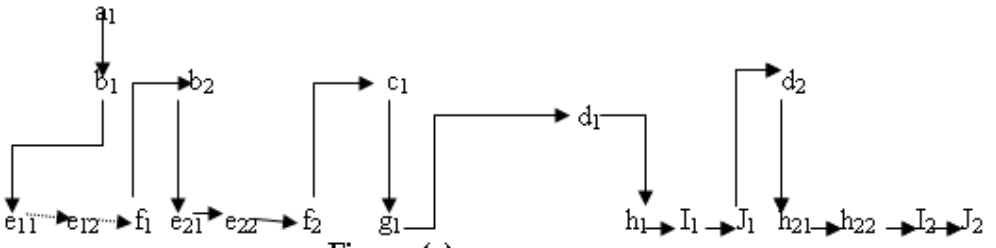


Figure (g)

Sample Database

In order to understand the hierarchical data model better, let us take the example of the sample database consisting of supplier, parts and shipments. The record structure and some sample records for supplier, parts and shipments elements are as given in following tables

The Supplier records

Sno	Name	Status	City
S1	Suneet	20	Qadian
S2	Ankit	10	Amritsar
S3	Amit	30	Amritsar

The Part records

Pno	Name	Color	Weight	City
P1	Nut	Red	12	Qadian
P2	Bolt	Green	17	Amritsar
P3	Screw	Blue	17	Jalandhar
P4	Screw	Red	14	Qadian

The Shipment records

Sno	Pno	Qty
S1	P1	250
S1	P2	300
S1	P3	500
S2	P1	250
S2	P2	500
S3	P2	300

We assume that each row in supplier table is identified by a unique SNo that uniquely identifies the entire row of the table. Likewise each part has a unique Pno. Also we assume that no more than one shipment exists for a given supplier/part combination in the shipments table.

Hierarchical view for the suppliers-parts database :

The following figure shows four individual trees , a tree for each part .

P3	Screw	Red	14	Gadian

S1	Saneet	20	Qadian	500

P1	Nut	Red	12	Qadian

S1	Suneet	20	Qadian	250
S2	Ankit	10	Amritsar	250

P2	Bolt	Green	17	Amritsar

S1	Suneet	20	Qadian	300
S2	Ankit	10	Amritsar	500
S3	Amit	30	Amritsar	300

P4	Screw	Red	14	Qadian

The tree structure has parts record superior to supplier record. That is parts from the parent and supplier forms the children. Each of the four trees in above figure consists of one part record occurrence, together with a set of subordinate supplier record occurrences. There is one supplier record for each supplier of a particular part. Each supplier occurrences include the corresponding shipment quantity. For example, supplier S3 supplies 300 quantities of part P2. Note that the set of supplier occurrences for a given part occurrence may contain any number of members, including zero (for the case of part P4). Part P1 is supplied by two suppliers, S1 and S2. Part P2 is supplied by three suppliers, S1, S2 and S3 and part P3 supplied by only supplier S1.

Operations on Hierarchical Model

There are four basic operations Insert, Update, Delete and Retrieve that can be performed on each model. Now we consider in detail that how these basic operations are performed in hierarchical database model.

Insert Operation: It is not possible to insert the information of the supplier e.g. S4 who does not supply any part. This is because a node cannot exist without a root. Since, a part P5 that is not supplied by any supplier can be inserted without any problem, because a parent can exist without any child. So we can say that insert anomaly exists only for those children, which has no corresponding parents.

Update Operation : Suppose we wish to change the city of supplier S1 from Qadian to Jalandhar , then we will have to carry out two operations i.e. searching S1 for each part and then performing updations for different occurrences of S1 .But if we wish to change the city of part P1 from Qadian to Jalandhar, then these problems will not occur because there is only a single entry for part P1 and the problem of inconsistency will not arise . So we can say that update anomalies only exist for children not for parent because children may have multiple entries in the database.

Delete Operation: In hierarchical model, quantity information is incorporated into supplier record . Hence the only way to delete shipment (or supplied quantity) is to delete the corresponding supplier record. But such an action will lead to loss of information of the supplier, which is not desired.

For example : Supplier S2 stops supplying 250 quantity of part P1 , then the whole record of S2 has to be deleted under part P1 which may lead to loss of information of supplier . Another problem will arise if we wish to delete a part information and that part happens to be only part supplied by some supplier. In hierarchical model, deletion of parent causes the deletion of child records also and if the child occurrence is the only occurrence in the whole database then the information of child records will also be lost with the deletion of parent. For example: if we wish to delete the information of part P2 then we also lose information of S3, S2 and S1 supplier. The information of S2 and S1 can be obtained from P1, but the information about supplier S3 is lost with the deletion of record of P2.

Record Retrieval: Record retrieval methods for hierarchical model are complex and asymmetric which can be clarified with the following queries:

Query 1: Find the supplier number for suppliers who supply part P2.

Solution: In order to get this information, first we search the information of parent P2 from database, since parent occurs only once in the whole database, so we obtain only a single record for P2 . Then, a loop is constructed to search all suppliers under this part and supplier numbers are printed for all suppliers.

Algorithm

```
get [next] part where PNO =P2;
do until no more shipments under this part;
get next supplier under this part;
print SNO;
end;
```

Query2: Find part numbers for parts supplied by supplier S2 .

Solution: In order to get required part number we have to search S2 under each part. If supplier S2 is found under a part then the corresponding part number is printed, otherwise we go to next part until all the parts are searched for supplier S2.

Algorithm

```
do until no more parts ;
        get next part;
        get [next] supplier under this part where SNO=S2;
        if found then print PNO;
end;
```

In above algorithm "next" is interpreted relative to the current position (normally the row most recently accessed; for the initial case we assume it to be just prior to the first row of the table). We have placed square brackets around "next" in those statements where we expect at the most one occurrence to satisfy the specified conditions.

1.9.4 Summary

The Network Data Model (NDM) represents data for any entity set by a logical record type. The data for an instance of the entity set is represented by a record occurrence of the record type. The DBTG model uses two different structures to represent the database entities and relationships between the entities, namely record type and set type. A record type is used to represent an entity type. It is made up of a number of data items that represent the attributes of the entity. A set type is used to represent a directed relationship between two records types, the so called owner record type, and the member record type. In the DBTG model, a data structure diagram consists of two record types and is linked to each other through many-to-one relationships. This structure is referred to as DBTG set. Bachman introduced a graphical means called a data structure diagram to denote the logical relationship implied by a set. Here a labeled rectangle represents the corresponding entity or record type. An arrow that connects two labeled rectangles represents the corresponding entity or record type. The arrow direction is from the owner record type to the member record type.

The hierarchical data model consists of a set of record types. The relationship between two record types is of the parent/child form, expressed using links or pointers. The records thus connected form an ordered tree, the so-called hierarchical definition tree.

The hierarchical model provides a straightforward method of implementing a one-to-many relationship. However, a many-to-many relationship between record types cannot be expressed directly in the hierarchical model. Such a relationship can be expressed by using data replication or virtual records.

The disadvantages of data replication are waste of storage space and the problem of maintaining data consistencies. A virtual record is a mechanism to point to an occurrence of a physical record. Thus, instead of replicating a record occurrence, single record occurrence is stored and virtual record points to this record wherever the record is required.

Comparison of Network and Hierarchical Model

S.No	Hierarchical Data Model	Network Data Model
1	Relationship between records is of the parent child type.	Relationship between records is expressed in the form of pointers or in the forms of links
2	Many to many relationship is not possible to represent in case of hierarchical data model	Many to many relationship can be very easily implemented in case of network data model.
3	It is a simple straight forward and natural method of implementing record relationship	Record relationship implementation is very complex due to the use of pointers.
4	This type of model is useful only when there is some hierarchical character in the database.	Network model is useful for representing such records which have many to many relationships.
5	In order to represent links among records, pointers are used. Thus relations among records are physical.	In network model also the record relations are physical.
6	Searching for a record is very difficult since one can retrieve a child only after going through its parent record.	Searching a record is easy since there are multiple access paths to a data element.
7	Hierarchical model suffers from **insert anomaly** as we cannot insert the information of a child who does not have any parent.	Network model does not suffer from **insert anomaly** as we can insert the information of a new supplier by showing a record set pointer to itself.

8	There are multiple occurrences of child records, which lead to problem of inconsistency during the **update** operation.	Network model is free from **update** anomalies because there is only a single occurrence for each record set.
9	It is based on parent child relationship and **deletion** of parent results in deletion of child records also.	It is based on many to many relationship (m:m) which make it free from **delete** anomalies .
10	Retrieve algorithms in case of hierarchical data model are complex and asymmetric.	Retrieve algorithms in case of network data model are complex and symmetric.

1.9.5 Self Understanding

1. What are features of Network data model?
2. Write short note on DBTG model.
3. Define DBTG Set, record type, set occurrence, set type, owner record type, and member record type.
4. List down the features of Hierarchical model.
5. Explain the ways of implementing the Hierarchical Data Model.
6. Explain the ways of implementing the Network Data Model.

1.9.6 Further Readings

Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.

C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.

Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

LESSON NO. 1.10 **AUTHOR : VISHAL GOYAL**

RELATIONAL DATA MODEL

Structure:

1.10.0 Introduction

The relational data model is an abstract theory of data that is based on certain aspects of mathematics(principally set theory and predicate logic).

The principles of relational model were originally laid down in 1969-70 by Dr. E.F. Codd at that time a member of IBM. Relational model is a way of looking at data. Relational model stores data in the form of tables. A relational model database is defined as a database that allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.

1.10.1 Objectives

After reading this lesson you will be able to

- Understand the basic concepts of Relational Data Model
- Constraints in Relational Data Model

1.10.2 Relational Data Model Concepts

The relational model is concerned with three aspects of data :

1) Structures
2) Data integrity
3) Manipulation

Structure aspects: The data in the database is perceived by the user as a table. It means database is arranged in the form of tables and collection of tables is called database. Structure means design view of database like data type, its size etc.

Integrity aspect: Those tables that satisfy certain integrity constraints like domain constraints, entity constraints, referential integrity and operational constraints.

Manipulative aspects: The operators available for the user for manipulating those tables into database e.g. for purpose of retrieval of data like projection, join and restrict.

Characteristics of Relational Database

Relational database system have the following characteristics:

1) The whole data is conceptually represented as an orderly arrangement of data into rows and columns called a relation or a table.
2) All values are scalar. That is, at any given time for each row/column position in the relation there is one and only one value.
3) All operations performed on an entire relation and result is an entire relation, a concept known as closure.

Dr Codd when formulating the relational model, chose the term "relation" because it was comparatively free of connotations, unlike, for example, the word "table". It is a common misconception that the relational model is so called because relationships are established between tables. In fact the name is derived from the relations on whom it is based. Notice that the model requires only that data be conceptually represented as a relation; it does not specify how the data should be physically implemented. A relation is a relation provided that it is arranged in row and column format and its values are scalar. Its existence is completely independent of any physical representation.

Emp_Code	Name	Year
21130	Amar Jain	1
30143	Kuldeep	3
41894	Manoj	2
51207	Rita Bajaj	6

Basic Terminology used in Relational Model

Tuples of a relation

Each row of data is tuple. Actually, each row in n-tuple, but the "n-" is actually dropped.

Cardinality of a relation

The number of tuples in a relation determines its cardinality. In this case, the relation in above figure has a cardinality of 4.

Degree of a relation

Each column in the tuple is called an attribute. The number of attributes in a relation determines its degree. The relation in above figure has degree 3.

Domains

A domain definition specifies the kind of data represented by the attribute. More particularly, a domain is a set of all possible values that an attribute may validly contain.

Domains are often confused with data types, but this is inaccurate. Data type is a physical concept while domain is a logical one. "Number " is a data type and "Age" is a domain .

To give another example "Street name" and "Sur name" might both be represented as text fields but they are obviously different kinds of text fields, they belong to different domain.

Domains is also a broader concept than data type in that domain definition includes a more specific description of the valid data. For example, the domain Degree Awarded, which represents the degrees awarded by a university in the database schema, this attribute might be defined as Text[3[, but it's not just any three–character string, it's a member of the set {BS, BA, MA, MS, PhD, LLB, MD}, of course not all hundred or so values if we are talking about mauseums exhibit. In such instances it's useful to define the domain in terms of the rules, which can be used to determine the membership of any specific value in the set of all valid values.

For example, Person Age could be defined as "an integer in the range 0 to 120" whereas Exhibit Age (age of any object for exhibition) might simply by "an integer equal to or greater than 0".

Body of a Relation

The body of the relation consists of an unordered set of zero or more tuples. There are some important concepts here. First the relation is unordered. Record numbers do not apply to relations. Second a relation with O tuples still qualifies as a relation. Third, a relation is a set. The items in a set are, by definition uniquely identifiable. Therefore for a table to qualify as a relation each record must be uniquely identifiable and the table must contain no duplicate constraints.

Keys of a Relation

It is a set of one or more columns whose combined values are unique among all occurrences in a given table. A key is the relational means to specify uniqueness.

1.10.3 Constraints :

Constraints are used to validate data entered for the specified column(s) namely :

There are two types of constraints

1) Table Constraint
2) Column Constraint

Table Constraint

If the constraint spans across multiple columns , the user will have to use table level constraints . If the data constraint attached to a specific cell in a table references the contents of another cell in the table , then the user will have to use table level constraints.

Primary Key as table level constraint :

E.g Create table sales_order_details(s_order_no varchar2(6), Product_no varchar2(6), PRIMARY KEY (s_order_no, product no));

Column Level Constraint

If the constraints are defined with the column definition, it is called as a column level constraint. They are local to a specific column.

Primary Key as a column level constraint

Create table client (client_no varchar2(6) Primary Key…);

Features of Constraint

1) NOT NULL CONDITION
2) UNIQUENESS
3) PRIMARY KEY identification

4) FOREIGN KEY

5) CHECK the column value against a specified condition

Some important constraints features and their implementation have been discussed below

Primary Key Constraints

A PRIMARY KEY constraint designates a column or combination of columns as the table's primary key. To satisfy a PRIMARY KEY constraint, both the following conditions must be true :

1) No primary key value can appear in more than one row in the table.

2) No column that is the part of the primary key can contain null value.

A table can have only one primary key.

A primary key column cannot be of data type LONG OR LONG ROW. You cannot designate the same column or combination of columns as both a primary key and a unique key or as both a primary key and a cluster key. However, you can designate the same column or combination of columns as both a primary key and a foreign key.

Defining Primary Keys

You can use the column_constraint syntax to define a primary key on a single column.

Example

The following statement creates the DEPT table and defines and enables a primary key on the DEPTNO column :

```
CREATE TABLE  dept
( deptno NUMBER (2) CONSTRAINT pk_dept PRIMARY KEY,
dname  VARCHAR2(10));
```

The pk_dept constraint identifies the deptno column as the primary key of the dept table . This constraint ensures that no two departments in the table have the same department number and that no department number is NULL .

Alternatively, you can define and enable this constraint with table constraint syntax :

```
CREATE TABLE dept
(deptno NUMBER(2),
dname  VARCHAR2(9),
loc VARCHAR2(10),
constraint PK_DEPT PRIMARY KEY (deptno));
```

Defining Composite Primary Keys

A composite primary key is a primary key made up of a combination of columns. Because Oracle creates index on the columns of a primary key, a composite primary key can contain a maximum of 16 columns. To define a composite primary key, you must use the table_constraint syntax, rather than the column_constraint syntax.

Example

The following statement defines a composite primary key on the combination of the SHIP_NO and CONTAINER_NO columns of the SHIP_CONT table :

```
ALTER TABLE ship_cont
        ADD PRIMARY KEY(ship_no, container_no) DISABLE
```

This constraint identifies the combination of the SHIP_NO and CONTAINER_NO columns as the primary key of the SHIP_CONTAINER. The constraint ensures that no two rows in the table have the same values for both SHIP_NO column and the CONTAINER_NO column.

The CONSTRAINT clause also specifies the following properties of the constraint .

1) Since the constraint definition does include a constraint name, Oracle generates a name for the constraint .
2) The DISABLE option causes Oracle to define the constraint but not enforce it.

Referential Integrity Constraints

A referential integrity constraint designates a column or combination of columns as a foreign key and establishes a relationship between that foreign key and a specified primary or unique key, called the referenced key. In this relationship, the table containing the foreign key is called the child table and the table containing the referenced key is called the parent table. Note the following:

1) The child and parent tables must be on the same database. They cannot on different nodes of a distributed database.
2) The foreign key and the referenced key can be in the same table. In this case, the parent and child tables are the same.
3) To satisfy a referential integrity constraint, each row of the child table must meet one following conditions:
    a) The value of the row's foreign key must appear as a referenced key value in one of the parent table's rows . The row in the child table is said to depend on the referenced key in the parent table.
    b) The value of one of the columns that makes up the foreign key must be null.

A referential integrity constraint is defined in the child table. A referential integrity constraint definition can include any of the following key words:

1) **F**oreign **K**ey **:** Identifies the column or combination of columns in the child table that makes up the foreign key . Only use this keyword when you define a foreign key with a table constraint clause.
2) **R**eference **:** Identifies the parent table and the column or the combination of columns that make up the referenced key. If you only identify the parent table and omit the column names, the foreign key automatically references the primary key on the parent table. The corresponding columns of the referenced key and the foreign key must match in number and data types.

On Delete Cascade : Allows deletion of referenced key values in the parent table that have dependent rows in the child table and causes Oracle to automatically delete dependent rows from the child table to maintain referential integrity . If you omit this Option, Oracle forbids deletions or referenced key in the parent table that have dependent rows in the child table.

In the first example, we defined a referential integrity constraint in a CREATE TABLE statement that contains as clause. Instead, you can create the table without the constraint and then add it later with an ALTER TABLE statement.

You can define multiple foreign keys in a table . Also , a single column can be part of more than one foreign key .

Defining Referential Integrity Constraints

You can use column_constraint syntax to define a referential integrity constraint in which the foreign key is made up of a single column .

Example

The following statement creates the EMP table and defines and enables a foreign key on the DEPTNO column that references the primary key on the DDPTNO column of the DEPT table :

```
CREATE TABLE emp
(Empno NUMBER (4) ,
ename VARCHAR2 (10) ,
job VARCHAR2 (9) ,
ngr NUMBER (4) ,
hiredate DATE ,
sl NUMBER (7,2) ,
deptno CONSTRAINT fk_deptno REFERENCES dept (deptno));
```

The constraint FK_DEPTNO ensures that all employees in the EMP table work in a department in the DEPT table. However, employees have null department numbers.

Before you define and enable this constraint you must define and enable a constraint that designates the DEPTNO column of the DEPT table as a primary or unique key .Note that the referential integrity constraint definition does not use the FOREIGN KEY keyword to identify the columns that make up the foreign key. Because the constraint is defined with a column constraint clause on the DEPTNO column, the foreign key is automatically on the DEPTNO column.

Note that the constraint definition identifies both the parent table and the columns of the referenced key. Because the referenced key is the parent table's primary key, the referenced key column names are optional.

Note that the above statement omits the DEPTNO column's data type. Because this column is a foreign key, Oracle automatically assigns it the data type DEPTNO column to which the foreign key refers.

Alternatively, you can define a referential integrity constraint with table_constraint syntax :

```
CREATE TABLE emp
```

(empno NUMBER (4) ,
ename VARCHAR2 (10) ,
job VARCHAR2(9) ,
ngr VARCHAR2(9) ,
Hiredate DATE ,
Sl NUMBER(7,2) ,
Comm NUMBER(7,2) ,
Deptno CONSTRAINT fk_deptno FOREIGN KEY
(deptno) REFERENCES dept(deptno));

Note that the foreign key definitions in both the above statements omit the ON DELETE CASCADE option , causing Oracle to forbid the deletion of a department if any employee works in that department .

Now if we take a simple example with following relations based on which we see the various operations in relational model. Relations are

1) Supplier records
2) Part records
3) Shipment records

The Supplier records

SNo	Name	Status	City
S1	Suneet	20	Qadian
S2	Ankit	10	Amritsar
S3	Amit	10	Amritsar

The Part records

PNo	Name	Color	Weight	City
P1	Nut	Red	12	Qadian
P2	Bolt	Green	17	Amritsar
P3	Screw	Blue	17	Jalandhar
P4	Screw	Red	14	Qadian

The Shipment records

Sno	Pno	Qty
S1	P1	250
S1	P2	300
S1	P3	500
S2	P1	250
S2	P2	500
S3	P2	300

As we discussed earlier , in this context we assume that each row in the Supplier table is identified by a unique SNo (Supplier Number), which uniquely identifies the entire

row in the table . Likewise each part has a unique PNo (Part Number) .Also we assume that no more than one shipment exists for a given supplier/part combination in the shipments table .

Note that the relations Parts and Shipments have PNo (Part Number) in common and Supplier and Shipments relations have SNo (Supplier Number) in common. The Supplier and Parts relation have City in common. For example, the fact that supplier S3 and part P2 are located in the same city is represented by the appearance of the same value, Amritsar, in the city column of the two tuples in relations.

Operations in Relational Model

There are four basic operations :

1) Insert
2) Delete
3) Modify
4) Retrieve

Insert Operation :

Suppose we wish to insert the information of supplier who does not supply any part, it can be inserted in S table without any anomaly e.g. S4 can be inserted in S table. Similarly, if we wish to insert information of a new part that is not supplied by any supplier, it can be inserted into a P table. If a supplier starts supplying any new part, then this information can be stored in shipment table SP with the supplier number, part number and supplied quantity. So we can say that insert operations can be performed in all the cases without any anomaly.

Modify Operation

Suppose supplier S1 has moved from Qadian to Jalandhar. In that case we need to make changes in the record so that the supplier table is up-to-date. Since supplier number is the primary key in the S table. so there is only a single entry of S1, which needs a single update and problem of data inconsistencies would not arise. Similarly, part and shipment information can be updated by a single modification in the tables P and SP respectively without the problem of inconsistency. Update operation in relational model is very simple and without any anomaly in case of relational model.

Delete Operation:

Suppose if supplier S3 stops the supply of part P2, then we have to delete the shipment connecting part P2 and supplier S3 from shipment table SP. This information can be deleted from SP table without affecting the details of supplier of S3 in supplier table and part P2 information in part table. Similarly, we can delete the information of parts in P table and their shipments in SP table and we can delete the information of suppliers in S table and their shipments in SP table.

Record Retrieval:

Record retrieval methods for relational model are simple and symmetric which can be clarified with the following queries :

Query 1: Find the supplier numbers for suppliers who supply part P2

In order to get this information we have to search the information of part P2 in the SP table. For this a loop is constructed to find the rcords of P2 and on getting the records, corresponding supplier numbers are printed.

Algorithm

```
do until no more shipments;
get next shipment where PNO=P2;
print SNO;
end;
```

Query 2 : Find part numbers for parts supplied by supplier S2.

In order to get this information we have to search the information of supplier S2 in the SP table. For this a loop is constructed to find the records of S2 and on getting the records corresponding part numbers are printed.

Algorithm

```
do until no more  parts ;
get next shipment where SNO=S2;
        print PNO;
end;
```

Since both the queries involve the same logic and are very simple, so we can conclude that retrieval operation of this model is simple and symmetric.

Structured Query Language (SQL)

Structured query language (SQL) pronounced as "sequel" is the set of commands that all programs and users must use to access data within the database. Application programmers and Oracle tools often allow users to access the database without directly using SQL, but these applications in turn must use SQL when executing the user's request.

Historically, the paper, " A Relational Model of Data for Large Shared Data Banks," by Dr E F Codd, was published in June 1970 in the Association of Computer Machinery (ACM) journal, Communications of the ACM. Codd's model is now accepted as the definitive model for relational database management systems (RDBMS). The language, Structured English Query Language (SEQUEL) was developed by IBM Corporation, Inc .to use Codd's model. SEQUEL, later became SQL. In 1979, Relational Software, Inc introduced the first commercially available implementation of SQL. Today, SQL is accepted as the standard RDBMS language. The latest SQL standard published by ANSI and ISO is often called SQL-92 (and sometimes SQL2).

Benefits of SQL

This section describes many of the reasons for SQL's widespread acceptance by relational database vendors as well as end users. The strengths of SQL benefit all ranges of users including application programmers, database administrators, and management and end users.

Non-Procedural Language

SQL is a non-procedural language because it :

1) Processes sets of records rather than just one at a time ;
2) Provides automatic navigation to the data .
3) System administrators
4) Database administrators
5) Security administrators
6) Application programmers
7) Decision support system personnel
8) Many other types of end users

Unified Language

SQL provides commands for a variety of tasks including :

1. Querying data;
2. Inserting, updating and deleting rows in a table ;
3. Creating ,replacing , altering and dropping objects ;
4. Controlling access to the database and its object ;
5. Guaranteeing database consistency and language .

SQL unifies all the above tasks in one consistent language .

Common Language for all Relational Databases

Because all major relational database management systems support SQL, you can transfer all skills you have gained with SQL from one database to another. In addition, since all programmes written in SQL are portable, they can often be moved from one database to another with very little modification.

Embedded SQL

Embedded SQL refers to the use of standard SQL commands embedded within a procedural programming language. Embedded SQL is a collection of these commands:

All SQL commands, such as SELECT and INSERT, available with SQL with interactive tools;

Flow control commands, such as PREPARE and OPEN, which integrate the standard SQL, commands with a procedural programming language.

The Oracle precompilers support embedded SQL. The Oracle precompilers interpret embedded SQL statements and translate them into statements that can be understood by procedural language compilers. Each of these Oracle precompilers translate embedded SQL programmes into a different procedural language:

The Pro *Ada precompiler
The Pro *C/C++ Precompiler
The Pro * COBOL precompiler
The Pro * FORTRAN precompiler
The Pro * Pascal precompiler
The Pro * PL/I precompiler

Database Objects :

Oracle supports two types of data objects .

Schema Objects : A schema is a collection of logical structures of data, of schema objects. A schema is owned by a database user and has the same name as the user. Each user owns a single schema. Schema objects can be created and manipulated with SQL and include the following types of objects.

Cluster	**database links**	**database triggers**
Indexes		**Packaged sequences**
Snapshots	**snapshot logs**	**shared functions**
Stored procedures	**synonyms**	**tables**
Views		

Non-schema Objects : Other types of objects are also stored in the database and can be created and manipulated with SQL , but are not contained in a schema .

Profiles	**Rates**
Rollback segments	**table spaces**
Users	

Objects Naming Conventions

The following rules apply when naming objects :

1) Names must be from 1 to 30 characters long with the following exceptions :
2) Names of database are limited to 8 characters. Names of database links can be as long as 128 characters.
3) Names cannot contain quotation marks.
4) Names are not case sensitive.
5) A name must begin with an alphabetic character from your database character set unless surrounded by double quotation marks.
6) Names can only contain alphanumeric characters from your database character set and the character _,$ and #. You are strongly discouraged from using $ and #.
7) If your database character set contains multi-byte characters, it is recommended that each name for a user or a role contain at least one single-byte character.
8) Names of databases links can also contain periods (.) and ampersand &.
9) Columns in the same table or view cannot have the same name. However, column in different tables or views can have the same name.
10) Procedures or functions contained in the same package can have the same name, provided that their arguments are not of the same number and data types.

Creating multiple procedures of functions with the same name in the same package with different arguments is called overloading the procedure or function.

Objects Naming Guidelines

There are several helpful guidelines for naming objects and their parts :

1) Use full, descriptive, pronounceable names (or well-known abbreviations).
2) Use consistent naming rules.
3) Use the same name to describe the same entity or attributes across tables.
4) When naming objects, balance the objective of keeping names short and easy to use with the objective of making names as long and descriptive as possible. When in doubt, choose the more descriptive name because many people may use the objects in the database over a period of time. Your counterpart ten years from now may have difficulty understanding a database with names like PMDD instead of PAYMENT_DUE_DATE.
5) Using consistent naming rules helps understand the part plays in your application. One such rule might be to begin the names of all tables belonging to the FINANCE application with FIN_.
6) Use the same names to describe the same things across tables. For examples, the department number columns of the EMP and DEPT tables should be named DEPTNO.

Advantages and Disadvantages of Relational Model

The major advantages of the relational model are :

Structural independence

In relational model changes in the database structure do not affect the data access. When it is possible to make change to the database structure without affecting the DBMS's capability to access data, we can say that structural independence has been achieved. So, relational database model has structural independence.

Conceptual simplicity:

We have seen that both the hierarchical and the network database models were conceptually simple. But the relational database model is even simpler at the conceptual level. Since the relational data model frees the designer from the physical data storage details, the designers can concentrate on the logical view of the database.

Design, implementation, maintenance and usage case:

The relational database model achieves both data independence and structure independence making the database design, maintenance, administration and usage much easier than the other models.

Ad hoc query capability:

The presence of very powerful, flexible and easy-to-use query capability is one of the main reasons for the immense popularity of the relational database model. The query language of the relational database models structured query language or SQL makes ad hoc queries a reality. SQL is a fourth generation language. A 4GL allows the user to specify what must be done without specifying how it must be done. So, using SQL the

users can specify what information they want and leave the details of how get the information to the database.

Disadvantages of Relational Model

The relational model's disadvantages are very minor as compared to the advantages and their capabilities far outweigh the shortcomings . Also the drawbacks of the relational database systems could be avoided if proper corrective measures are taken. The drawbacks are not because of the shortcomings in the database model, but the way it is being implemented.

Some of the disadvantages are :

Hardware overheads:

Relational database system hides the implementation complexities and the physical data storage details from the users. For doing this, for making things easier for the users, the relational database systems need more powerful hardware computers and data storage devices. So the RDBMS needs powerful machines to run smoothly. But as the processing power of modern computers is increasing at an exponential rate and in today's scenario, the need for more processing is no longer a very big issue.

Ease of design can lead to bad design

The relational database is an easy to design and use. The users need not know the complex details of physical data storage. They need not know how the data is actually stored to access it. This ease of design and use can lead to the development and implementation of very poorly designed databases. Since the database is efficient, these design inefficiencies will not come to light when the database is designed and when there is only a small amount of data. As the database grows, the pooly designed databases will slow the system down and will result in performance degradation and data corruption.

Information island phenomenon

As we have said before, the relational database systems are easy to implement and use. This will create a situation where too many people or departments will create their own databases and applications.

These information islands will prevent the information integration that is essential for the smooth and efficient functioning of the organization. These individual databases will also create problems like data inconsistency, data duplication, data redundancy and so on.

But as we have said all these issues are minor when compared to the advantages and all these issues could be avoided if the organization has a properly designed database and has enforced good database standards.

1.10.4 Summary

The relational data model was first introduced by Dr. E.F. Codd, an Oxford trained mathematician while working in IBM Research Center in 1970's. He represented this idea in a classic paper and attracted immediate attention due to its simplicity and mathematical foundations. It also drew immediate attention of the computing industry because of its simple way in which it represented information by well understood convention of tables of values as its building block. The relational model is considered one

of the most popular developments in the database technology because it can be used for representing most of the real world objects and the relationships between them.

1.10.5 Self Understanding

Q1. Explain the various concepts of relational data model.

Q2. Explain the various constraints in relational data model.

1.10.6 Further Readings

Bipin C. Desai, *An introduction to Database System,* Galgotia Publication, New Delhi.

C. J. Date, An *introduction to database Systems*, Sixth Edition, Addison Wesley.

Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of Database Systems*, Addison Wesley.

Type Setting :
Department of Distance Education, Punjabi University, Patiala.