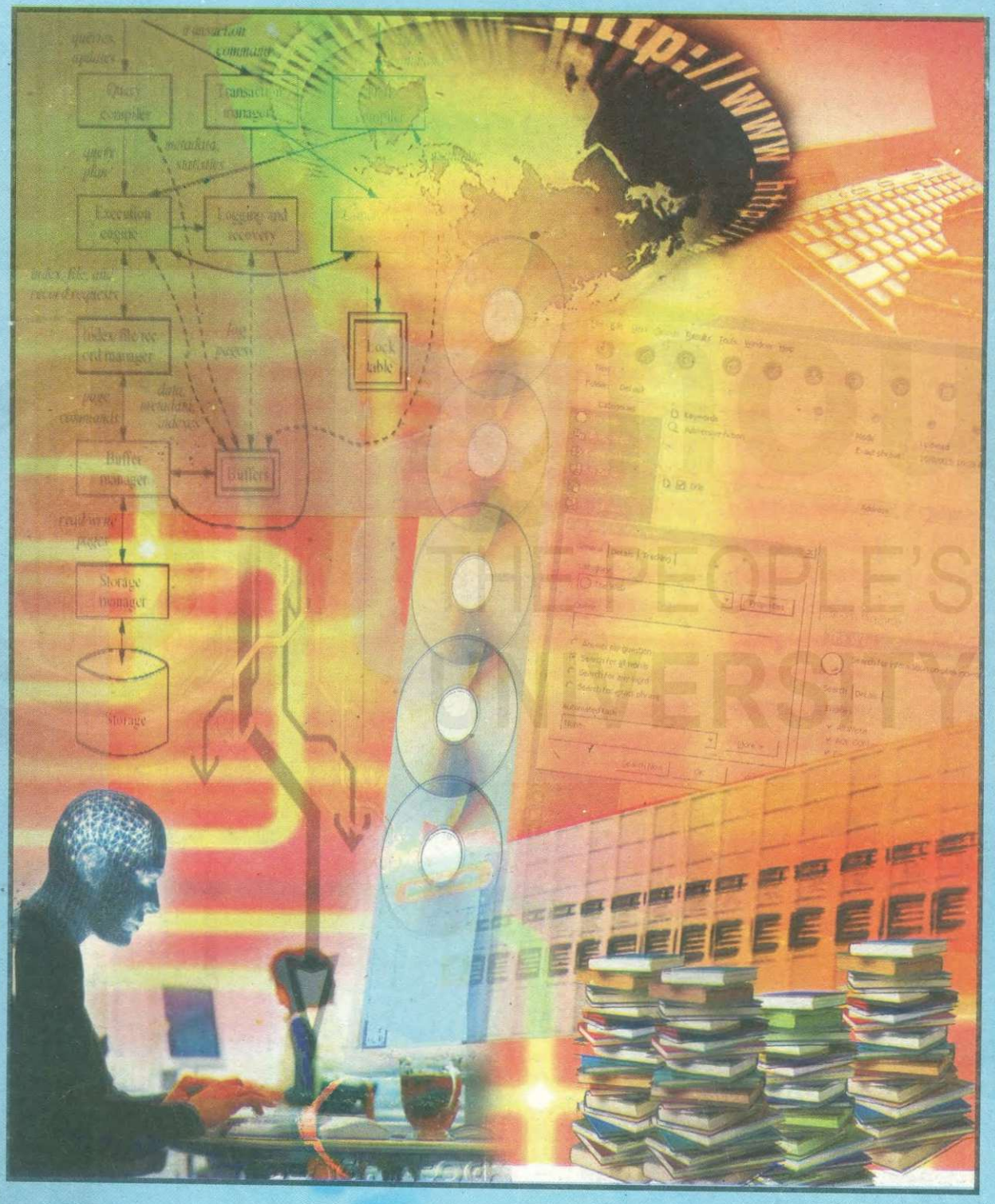
Indira Gandhi National Open University
School of Social Sciences

MLII-104

Master's Degree in Library and Information Science (MLIS)



INFORMATION COMMUNICATION TECHNOLOGIES: APPLICATIONS

PART-I

"शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्रा की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।"

- इन्दिरा गाँधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

- Indira Gandhi

Indira Gandhi
National Open University
School of Social Sciences

MLII-104

Information Communication Technologies: Applications

Programme Design Committee (Original) (1992)

Prof. Pandav Nayak
(Chairman)
Prof. (Ms.) A. K. Anand
Prof. J. C. Binwal
Prof. M.A. Gopinath
Prof. B. Guha
Prof. S. R. Gunjal
Dr. S. G. Mahajan
Shri N. M. Malwad
Dr. S. S. Murthy
Prof. K.S. Raghavan
Prof. T. N. Rajan
Prof. A.P. Srivastava
Prof. T. Viswanathan
Dr. R. Satyanarayana *(Convener)*
Dr. (Ms.) Neela Jagannathan
(Special Invitee)
Dr. Uma Kanjilal
(Internal Faculty)
Ms. Neena Talwar Kanungo
(Internal Faculty)

Course Preparation Team (1992)

Prof. M.A. Gopinath
Dr. L. J. Haravu
Dr. Uma Kanjilal
Course Editor : Prof. T. N. Rajan
Prof. T. N. Rajan
Prof. M.R. Riswadkar
Dr. R. Satyanarayana
Prof. A.P. Srivastava

Programme (Curriculum) Revision Committee (2003)

Prof. S.B. Ghosh (Chairman)
Faculty of Library and
Information Science
IGNOU, New Delhi

Dr. Jagdish Arora
Chief Librarian
IIT, Delhi

Prof. S.R. Ganpule
Dept. of Library &
Information Science
Yaswant Rao Chavan
Maharashra Open University
Nasik

Prof. B. Guha
Professor (Retd.)
Dept. of Lib. & Inf. Sc.
Banaras Hindu University
Banaras, U.P.

Prof. Uma Kanjilal
Faculty of Library and
Information Science
IGNOU, New Delhi

Dr. Neena Talwar Kanungo
Faculty of Library and
Information Science
IGNOU, New Delhi

Prof. C.R. Karisiddappa
Head, Dept. of Library &
Information Science
Karnataka University
Dharwar

Dr. A. Lahiri
Senior Advisor
Dept. of Scientific &
Industrial Research
New Delhi

Prof. P.B. Mangla
Deptt. of Library &
Information Science (Retd)
University of Delhi
Delhi

Prof. A. Neelameghan
UNESCO Expert (Retd.)
Sarada Ranganathan
Endowment for Library &
Information Science
Bangalore

Prof. T.N. Rajan
Chief Coordinator (Retd.)
INSDOC, New Delhi

Dr. Pravakar Rath
Faculty of Library and
Information Science
IGNOU, New Delhi

Prof. J. Sarkhel
Head, Dept. of Library &
Information Science
Vidyasagar University
Midnapur
West Bengal

Prof. R. Satyanarayana
Faculty of Library and
Information Science
IGNOU, New Delhi

Prof. B.K. Sen
INSDOC (Retd.)
New Delhi

Dr. Mahinder Singh
Director
DESIDOC, Delhi

Dr. N. Vijayaditya
Director General
National Informatics Centre
New Delhi.

Prof. T. Viswanathan
Ex-Director, INSDOC
New Delhi

Dr. (Ms.) Neela Jagannathan
(Special Invitee)
IGNOU, New Delhi

Programme Coordinators : Prof. S.B. Ghosh and Prof. Uma Kanjilal

Course Coordinator : Dr. Jaideep Sharma

Course Editor : Dr. S.S. Murthy

Course Preparation Team

Unit	Contributor
1	Prof. R. Satyanarayana
2-3	Mr. J. M. Bhardwaj
4	Prof. K. S. Raghavan
5-6	Mr. P. S. Mukherjee
7,12-14	Dr. Jagdish Arora
8-9	Dr. P. Vyasamoorthy
10-11	Dr. J. P. Singh
15-16	Dr. Usha Mujoo-Munshi

Secretarial Assistance : Mr. Rajiv Kumar

Material Production

Miss Pushpa Gupta
Mr. Arvind Kumar

Mrs. M. Sumathy Nair

December, 2008 (Reprint)

ISBN : 81-266-2078-1

Further information on the Indira Gandhi National Open University courses may be obtained from the University's Office at Maidan Garhi, New Delhi-110 068

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi, by Director, School of Social Sciences.

"Paper Used: Agrobased Environment Friendly"

Laser Typeset by : Rajshree Computers, V-166A, Bhagwati Vihar, Uttam Nagar, New Delhi-110059.

Printed at A-One Offset Printers, 5/34 Kirti Nagar Indl. Area, New Delhi-110 015.

ignou
THE PEOPLE'S
UNIVERSITY

COURSE INTRODUCTION

MLIL-104 titled "Information Communication Technologies: Applications" has the objective of introducing the learners to the various aspects of ICT so as to enable them to apply these aspects in library and information services. After completing this course, they will acquire the competence to create databases of the holdings of their libraries as well as digitise the resources.

This course is divided into four Blocks. Each Block is further divided into four units each making a total of sixteen units. The Blocks are titled as:

- Block 1: Database Design and Management
- Block 2: Library Automation
- Block 3: Library and Information Services
- Block 4: Internet Resources and Services

Block-1 consists of Unit Nos. 1-4. It introduces you to the concept of database. There is a discussion on the difference between database and file structure and database design. You shall come to know the difference between traditional file structure and database design, its advantages over the former. The Block provides a detailed knowledge of database design and searching.

Block-2 is devoted to library automation. After knowing the importance of DBMS, designing and management of data in databases in the first block, the second block introduces the learners to automate the activities performed in their libraries and digitise the resources. Unit-5 of the Block is on library housekeeping operations that have been defined as the activities undertaken in the library to provide services. These activities start from acquisition to processing of the documents and finally maintaining them. Such activities demand a lot of time and energy and are routine and repetitive in nature. Their automation is important to systematise the working; make the processes more effective and efficient; and relieve the staff from routine activities for providing innovative services. Unit – 6 provides a detailed discussion on library software packages, which includes their generations, functions, general requirements, and the implementation in libraries. Thereafter a number software packages, both Indian and foreign have been discussed at length in this Unit. Unit - 7 is on digitisation which is the most recent and important topic in library automation today. It is important to digitise the resources of the library in view of the trend today towards e-resources and digital libraries. This Unit describes the process, technology, planning and implementation of the process of digitisation. It would help the learners to undertake digitisation of their libraries in the future.

Unit Nos. 9-12 of Block-3 of the course elaborate on the services viz., Alerting Services, Bibliographic Full Text Services, Document Delivery Services, and Reference Services. CAS, SDI have been discussed in Alerting Services. Examples of various services and databases available have been given. Electronic News Clipping Services and Full Text Services have been explained. The concept, need and players involved in full text services have been described. Library consortia, e-print archives, and full - text resources available on the Internet also get proper mention here. Copyright, another important issue in full text resources has also been discussed. Unit-11 is on Reference Services; it covers digital reference service need and projects beside the reference process and query.

The course closes with Block-4 titled " Internet Resources and Services" with five units (Units 13-16), viz. Basics of Internet, Use of Search Engines, Internet Services, Internet Information Resources, and Evaluation of Internet Resources. Internet has made a tremendous effect on libraries and thereby the services rendered by them. A study of Internet fundamentals, its resources and services is essential for any librarian today. He/She has to adapt and innovate his/her services utilising the vast resources available on the Internet. Internet is a medium free for all; anyone can publish anything on it instantaneously made accessible to everyone. The information available on the net has to be used with caution. The last Unit on Evaluation of Internet Resources, explains the need and process of Internet Information resources to verify their authenticity and reliability.

BLOCK 1

DATABASE DESIGN AND MANAGEMENT

UNIT 1 DATABASE: CONCEPT AND COMPONENTS

Structure

1.0 OBJECTIVES

After reading this Unit, you will be able to:

- understand as to what database approach is and trace the evolutionary pattern of a database;
- comprehend the meaning of the term database and define it;
- distinguish between a conventional file structure and a database;
- identify the major components of a bibliographic database along with their functions; and
- know the different types of bibliographic databases and their uses.

1.1 INTRODUCTION

Data is an important resource for any organisation. Many organisations/enterprises irrespective of their nature are concerned with collection and manipulation of data. Data can be defined as the raw numbers or letters stored in a computer, which represents facts about things. Data has little meaning in its stored state, whether we think of data in the physical form in which it is stored or in its logical form, since facts without context are unusable. A store of data is referred to as a database.

Ideally, data must be accessible to any authorised person in an organisation for whatever application in whatever format desired. But, traditional computer files maintained in different organisations do not facilitate the access of this nature. In other words, computer files are designed to be of use for just one application and the set of programs associated with that application. They generally have little or no relation to other data used and required

within the same organisation because they are initiated and developed independently. Such files are often unrelated to user's requests for information, particularly when these requests change in un-predictable ways.

When files are designed for a specific application, and when an organisation has many applications, the files often contain redundant data. This means the same data is stored in more than one file. For example, an employee's name and address might be contained in both personnel and payroll files. In addition to requiring more storage space, redundant data presents updating problems, since updating data in one file will not update the same data in another file. As a result, data in one file may not correspond to data in another file, although, it is supposed to. Further, since files are designed for application specific programs, a change in the format of data can affect many applications and many programs. If a new field has to be added to a file, then the associated program must be changed. In other words, it might be said that the *programs are data dependent.*

To overcome the problems of data unrelated to user's needs, data redundancy, and *program dependency on data,* databases have been developed. The basic purpose of a database is to enable access to stored data so that it can provide information that users require. This Unit is intended to furnish detailed information relating to database concept and explain the database components.

1.2 DATABASE APPROACH

Traditionally, data accessed through computers has been stored on different storage media in the form of individual files. Files proved to be quite satisfactory so long as computerisation was limited to a few application areas and use of computers is restricted. However, as the actual users grew in number, with the advent of online sharing systems, the file systems gave rise to many serious problems. The discipline of database systems evolved in response to these problems.

It may be emphasised that the database approach is more than merely a different computer technique involving the storage of data and the use of additional generalised software. It involves a new approach to designing and operating information systems throughout the enterprise and can have far reaching effects beyond the data processing department. Stated differently, database approach regards data as a resource to be managed along with more generally recognised resources of an organisation (such as staff, finance, equipment and premises) so as to be available to a variety of applications and users. The integrated database is intended to provide a consistent view of the institution's/enterprise's data for all user departments. Although, each department has responsibility for specific data, several departments using the same type of data can operate on the same data values. In other words, a parochial view of data would be discouraged and the uncoordinated exchange of information among departments would not be necessary. Instead, all basic data would be input to the database by each department responsible for it and retrieved (possibly in summarised form) by those departments needing access to it.

The data base approach is not dependent on any particular structure of data; many of these concepts can be applied to files as well. The database approach cannot be practically implemented in the absence of a computer. In many respects database approach is characterised by the word efficiency. In principle, one can do any thing with non-database systems as with database systems. In practice, the cost of doing certain things without the benefit of the database approach is so prohibitive that the possibility is dismissed. The two main contributors to prohibitive cost are the data redundancy and the uncontrolled searching time. The database approach is beneficial when online access to data is provided. With online access, the collection of data and extracting of timely reports becomes fairly straightforward as the restrictions or delays of batch processing are not imposed on the user. However, it may be emphasised that the database approach is rooted in the attitude of:

- sharing valued data resources;
- releasing control of those resources to a common responsible authority; and

- cooperating in the maintenance of those shared data resources.

This approach becomes significant when one is aware of the limitations of traditional computer files. However, implementation of this approach calls for a phased programme.

Database approach is diagrammatically represented in the following figure:

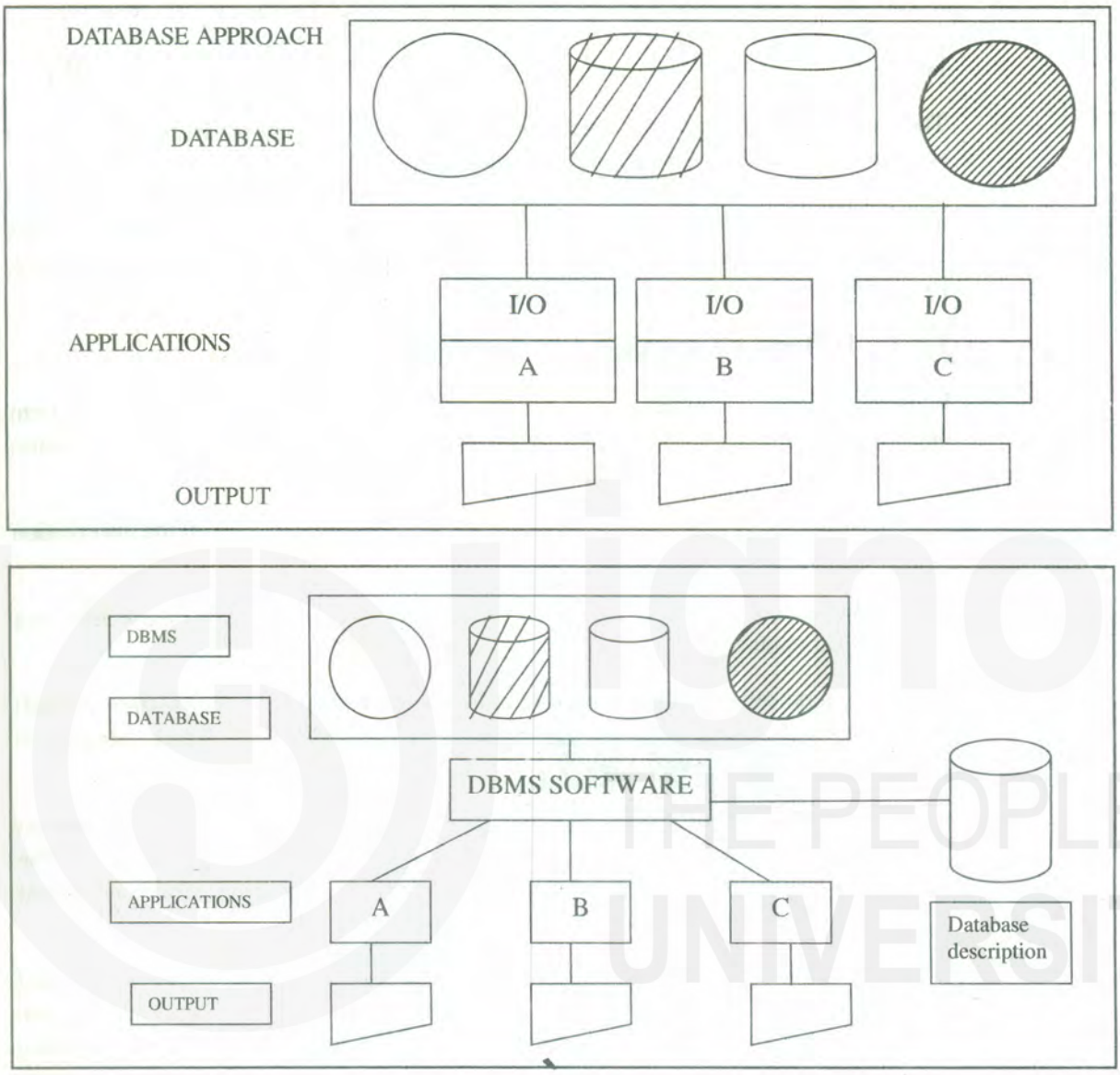


Fig. 1: Database Approach

Self Check Exercise

1) What is database approach? Enumerate the attitudinal factors that contribute to database approach.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit

.....

.....

.....

.....

.....

1.3 DATABASE DEFINITION

The major problem one faces in the study of database technology relates to the determination as to what precisely constitutes a database. The casual use of the term database tends to refer to any organised collection of data capable of being accessed by a computer. This could be applied to a couple of reels of magnetic tape or a few boxes of punched cards (obsolete now) or a collection of floppies containing data. As such, this interpretation does not constitute a precise definition for the concept.

The literature published on the subject of databases in the field of library and information science tends to focus on the use of online search services and searching techniques. There is not much clearly identifiable literature on the creation and management of textual databases covering the procedural aspects of building and managing information packages consisting largely bibliographic information. This section outlines the concept of database, issues related to its need and its composition.

1.3.1 Different Approaches to Database

In the literature of computer science, one encounters a number of definitions for the term database. Let us examine some of the definitions so as to understand different approaches to this concept.

i) "a collection of data on a defined range of subjects together with all the information needed to access that data".

ii) "a named collection of units of physical data which are *related to each other* in a specific manner".

iii) "a generalised, common *integrated collection* of (company or installation owned) data which fulfils the data requirements of all applications which exist in an enterprise".

iv) "the term database refers to just *the information file.* Database software is the set of programs whose function is *to manage the data and programs that operate on it.* The database system is *the entire hierarchy of elements, files and application programs* that result in efficient management of information".

v) "a database is a *collection of data organised in a manner which allows retrieval and uses that data by any one needing it.* A database is organised and designed *to allow a large number of users to draw information from it for many purposes in many different formats".*

vi) "in its most basic form, a database consists of a number of data elements, each of which is a unit of data *that is complete in it self.* A part number for example, is a typical *data element.* These elements are organised into *logically related groups* called *data structures....* The data files in database system are organised in a fashion that permits their use in *several applications* rather than *a single application.* Thus, in a database system *the focus shifts* from a *particular application* and its *specific input and output needs to a more general requirement for the data files to serve a number of applications'.*

It may be noted that each of the above definitions regards the 'database' from a different viewpoint, its access, purpose, description, contents and integration. Yet, each refers to *a specific collection of organised data* rather than any data on computer-readable media. The last mentioned definition is a comprehensive one, which clearly explains the concept of a database and also brings out the essential difference between a database and a traditional computer file. This definition also refers to a database system. It is important to distinguish the term database from the term database system, all the components of which cooperate to collect, manipulate, manage and deliver information.

1.3.2 Database Features

One of the main purposes of a database is that the data in the database should be used for a variety of different applications. To achieve this it is important for a database to possess the following features:

- it must be substantially non-redundant (that is to say that the database should not have duplication of data) because duplication of data leads to difficulty in ensuring data consistency, and results in the wastage of storage space;
- it must be program-independent so that the data can be moved or restructured without the need to make alterations to programs. This concept is known as 'data indipendence';
- it must be capable of being used by all programs;
- it must include all necessary data relationships, to support the variety of different uses to which data is put;
- it must have common approach to retrieval, additions and deletions and amendments to data.

A database can be analysed from two viewpoints: the physical storage of the data or logical or conceptual view of data. Files are used to physically store data in a database. Most databases use either direct files or indexed files or a combination of the two to physically store data on disk. Users and applications do not need to know anything about the physical data storage. Stored with actual data will be a description of the database, which enables the DBMS to retrieve information from the database and to store new data in appropriate places in the database establishing relationships with other data if relevant.

The logical or conceptual database is concerned with how the data is logically organised and how the data can be retrieved for information purposes. In case access is required to a series of linked files, it is necessary to have guidelines regarding allocation of data to specific files within the database system, and defining the optimum links between files. Based on the model followed for the structuring of data there are three basic types of databases and the associated DBMS namely: hierarchical, network and relational.

Hierarchical databases are structured in such a way that the relationships between data items follow a branching tree-type arrangement. In other words, the database consists of elements, which act, in a *parent-child relationship*. The relationships within the database are established when the database is created, that is to say that the database designer defines which is a child element of the parent element. An element within a database can have only one parent element. The data stored in the lower levels of a hierarchy database can only be accessed through the parent element.

The network database approach is based on explicit links or pointers between related entities. In a network model of database, there is more direct link between the data items at various levels. This is achieved by the use of pointers linking data at different levels. This approach requires a large number of links established between data elements, which occupy a large amount of storage space.

Relational databases use a type of data structure, which has been commonly adopted in database systems. In relational database systems, information is held in a set of *relations* or in the form of *tables*. Rows in such tables correspond to *records* while columns in these tables are equivalent to *fields*. The data items in various relations are linked through a series of keys. Relational databases are designed using a technique known as 'normalisation'. 'Normalisation' is used to break data into *tables* so that the fields in each table are dependent only on one *key field* and *not linked to any other key*. This process ensures that insertions, deletions and amendments may be made on to the data without any difficulty.

In addition to the above-discussed types of databases, other types of database structures such as multimedia databases and object oriented databases are also in existence. Multimedia structures are used to manage such databases, which deal with pictures, animation, sound and text as well as tables. The storage needs of these materials are certainly different from the types discussed earlier.

The Multi Media DBMS (MM-DBMS) attempts to use a range of technologies like relational technology for tables, image storage devices for graphics and animation, and provide facilities to the users.

1.3.3 Databases in Library and Information Science

It must be mentioned here that the definitions of database discussed above are from the point of view of computer professional and are taken from the literature of computer science. However, within library and information science a database is defined (rather understood) as "an organised and generally unlinked set of machine-readable bibliographic or information source records". Taken collectively, these records constitute a growing file of information that can be used to obtain a variety of products for a range of purposes. These information files are defined according to their scope and subject coverage in many ways. The variety of database types has grown over the last two decades and includes specialist databases usually limited in scope to particular subject areas.

In order to understand the concept of bibliographic database, it is necessary to have some basic knowledge relating to files and records. A file is a collection of similar records, with relationship defined between the records. A record is the information contained in the database relating to *one document* or *item.* For example in a catalogue database, a record may contain all the information pertaining to a book. If it is a source database, a record may contain the contents of a directory entry or an article of a journal. Each record in a file is composed of a number of *fields.* Generally, there are two types of fields; *fixed length fields* and *variable length fields.* A fixed length field is one which contains the same *predetermined number of characters* in each record. Fixed length fields are easy and quick to code. They are ideal for such data elements as ISBNs and other type of information, which will be same in each record. On the other hand, a *variable length field* will consist of different lengths in different records. Variable length fields pose a problem for the computer as it cannot recognise where one field ends and other starts. This aspect needs a mechanism *to flag the beginnings* and *ends of the fields.* In other words, there should be codes to indicate *the start* as well as *the termination* of variable length fields. MARC record format provides such a mechanism. Normally, bibliographic databases use a mixture of fixed and variable length fields in order to accommodate the requirements of varying kinds of data. Also, within fields there can be units of information that could be designated as sub-fields. Sub-fields need to be flagged so that they can easily be identified.

This method of dividing records into fields and sub-fields allows sub-sets of the database to be selected and retrieved as per the query criterion. It will also facilitate partitioning of the database for the purposes of management and add precision for retrieval.

Clearly, what can be retrieved depends mainly on what is contained in it and the way in which information has been structured. The methods available for structuring of data have already been discussed under database features. One point to be added here, relates to inverted files. In the inverted file approach there may be two or three separate files. Generally, the two-file approach is commonly followed. This approach uses two files, namely, the text file and the index file. The text file contains the actual records while the index file provides access to these records. The index file contains a record for each of the indexed terms from all of the records in the database, arranged in an alphabetical order. Each term is accompanied by information on the frequency of its occurrence in the database in which it is to be located, the record in which it is entered. When a new record

is added to the database, it is necessary to update the index file. The text file and the index file are together used for database search. This principle is used in software packages like CDS/ISIS.

Another important concept, which needs to be understood in the context of bibliographic database, is *record format.* In order to facilitate exchange of bibliographic records between different computer systems, attempts have been made to develop standard record formats. Such formats have been found to be exceedingly useful in cataloguing applications. The standard record formats include the ones like MARC, UNIMARC and MARC-21, etc. These standard formats also embody agreements on the *elements* of a *bibliographic* record. The MARC format includes nearly *61 data elements,* of which 25 are directly searchable. It is compatible with the latest editions of AACR-2 and DDC. The MARC format is hospitable and can be modified to suit the varied needs. A detailed account of record formats is provided elsewhere. (Refer the Units 7 and 8, Block 2 of MLII-102).

To conclude this section on databases in Library and Information science (LIS), it may be stated that databases are usually defined in terms of information they contain and/or in terms of the attributes of system(s) used to update, manage, analyse, retrieve and display information from these databases. For example, DBMS generates databases, which have logically *consistent structures* in that records are actually linked and the information is shared and integrated in a single file for particular retrieval purposes. This distinction between the databases within library and information science and linked databases within DBMS reflects important differences related to the purposes each database is intended to serve.

1.3.4 Database Functional Considerations

The success of a database is largely determined by the amount and variety of information accessible and available to the users, and the facility available to users for submitting the requests and obtaining necessary responses. Data processing specialists frequently request the management a definition of their requirements for a specific database to be designed. The systems analyst assumes that a system cannot be designed until an itemised listing of user requirements and report formats have been prepared. In cases, where a singular specific application is involved, it may be possible to define each requirement in a detailed manner. However, when consideration is given to the needs of overall organisation, the requirement becomes more idealistic and complex. For the database to be dynamic in supplying information needs, it must also be dynamic and responsive to the changing information environment.

1.4 TYPES OF DATABASES

There are many ways of categorising databases. One of the categorisations might be a numerical and textual database. Another way of looking at them is by their coverage i.e., local, regional and global. Databases are generally stored on magnetic or optical media such as disks and accessed either locally or remotely. They may include access to a particular organisation's database covering transactions and financial records, or to other databases that might be accessed remotely. Some of these databases may hold publicly accessible information such as abstracting and indexing databases, full texts of reports or directories, etc., on the other hand, their might be databases which are shared within an organisation or group of organisations.

Databases available to information users in the public domain and which might be accessed remotely or online via some online search services, or locally on CD-ROM, can be categorised as *Reference* or *Source databases.*

Reference databases direct the user to another source where the information sought by the user is available. This might be a document, an organisation or an individual. Some of

the examples include: bibliographic databases, catalogue databases and referral databases. Let us try to know what these categories of databases are.

Catalogue databases

These types of databases indicate the collection of a given library or a group of libraries constituting a library network. These databases list the type of collection, namely, monographs, journal titles and other items possessed by the library. They merely provide citations to the documents along with their call numbers to enable easy location of the documents.

Referral databases

This type of databases offers references to information or data such as names and addresses of organisations, and other directory type of data.

It may be mentioned here that source databases contain the original source data, and are considered as one type of electronic document. After successful consultation of a source database, the user should have the information that is required by him and should not need to seek information in another original source. Source databases may be categorised into:

- numeric databases, which contain numerical data of different types such as statistics and survey data;
- full text databases of news paper items, technical specifications and computer software;
- hybrid databases which contain a mixture of textual and numeric data;
- multi-media databases, which include information stored in a mixture of different types of media including sound, video, pictures, text and animation.

Bibliographic databases

These types of databases contain citations or bibliographic references sometimes along with brief abstracts of literature. They indicate to the user the content of the full text, the source where it can be located (e.g. journal title, conference proceedings) and whether they contain mere citations or provide abstracts or summaries of the original documents covered.

Bibliographic databases, contain a series of bibliographic records, with each record containing some combination of the under mentioned components:

- document number
- author
- title
- source reference
- abstracts
- full text
- indexing terms or key words or phrases
- citations including the total number of references
- language of the document
- call number or location

Each of these items is known as a data element and is represented by a field. There are different bibliographic record formats and there is considerable variation between them. It may be mentioned here that the components listed above do not generally (except in case of those containing abstracts) give information of the text of the document but only indicate where the information might be found. Of course, a good informative abstract, if

provided for each reference may furnish valuable information to the user and enhance the utility of the database.

Table 1.1: Some Examples of Bibliographic Databases

Database	Producer	Content
ABI-INFORM	University Microfilm Inc.	Business and management
BIOSIS Previews	BIOSIS	Bioscience
CA	CAS (Chemical Abstracts)	Chemistry
COMPENDEX	Engineering Information Inc.	Engineering
DISSABS	University Microfilm Inc.	Bibliographic details and abstracts of dissertations submitted to North American and some other universities
EMBASE	Elsevier Science	Biomedicine and pharmaceuticals
EVENTLINE	Publishers Elsevier	Multidisciplinary conferences and events
INPADOC	Science Publishers	International patents
INSPEC	European Patent Office	Engineering, physics and electronics
JGRIP	Institution of Electrical Engineers	Multidisciplinary research from public research organisations in Japan Medicine
MEDLINE	The Japan Science and Technology Corporation	
SCISEARCH	US National Library of Medicine	
	Institute of Scientific Information	Science and technology, bibliographic with citation references

Table 1.2: Some Examples of Source Databases

Database	Producer	Content
EDOC	European Patent Office	Patent Applications
FMARK	Institut National de la Propri'ete' Industielle	Trade Marks registered in France
DUNS United Kingdom	Dun & Bradstreet	UK Companies Information
ASPO	Agence France-Press	French and International sports newswires, reports, results, summaries and biographies
LOGOS	Documentation Francaise	Full text information about French politics, society and economy
PROMT	Information Access Company	Articles on products, markets and technologies
(Who's Who in Technology)	Gale Research Inc.	Details of leaders of American technology
Gale Directory of Databases	Gale Research Inc.	Descriptions of databases available on CD-ROM, online and other formats

1.5 DATABASE ARCHITECHTURE

The architecture of a database is commonly viewed in terms of three separate levels of description: *conceptual, external* and *internal.*

The overall logical description of the entire database is *the conceptual level.* This overall description is commonly known as a *schema.* It may also be called a community user view. Subsets of the schema that contain only the data needed for particular applications may be defined. These are called *sub- schemas* or user views. The sub-schemas provide a description at the external level. The description of physical storage structures used to store database on a specific computer system is the internal description.

Explained in simple language, it might be stated that a database can be analysed from two view-points- the physical storage of the data and the logical or conceptual view of data. Files are used to physically store data in a database. Most databases use either direct files or indexed files or a combination of the two to physically store data on a disk.

The *logical* or conceptual view of a database is concerned with *how data is logically organised and how data can be retrieved* for information purposes. There are *three different methods* (architectures) of logically organising data in a database. They are hierarchy model, network model and relational model. These are explained in Unit 2 of this block.

Self Check Exercise

2) Define database in the context of library and information science.

3) Discuss in brief the different levels of database architecture.

Note: i) Write your answers in the space given below.

ii) Check your answers with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

1.6 SUMMARY

This Unit commences with an introduction in which the importance of data and its organisation has been emphasised. The term 'database' has been defined and interpreted in different ways in the literature of computer science. The Unit discusses the concept of 'database' and explains the reasons for its emergence. It also includes a detailed discussion on database approach. It has been emphasised that the database approach developed in an attitude of sharing valued data resources, releasing control of those resources to a common responsible authority cooperating in the maintenance of the shared data resources.

Some of the definitions of 'database' selected from published literature have been critically examined, analysed and the significant aspects of these definitions that would facilitate a better understanding of the database concept have been highlighted. The basic difference between a database and traditional computer files has been delineated and has been explained. In addition, attempt has been made to bring out in brief the distinction between the DBMS-based general databases and databases as perceived and understood in the field of Library and Information Science. Different types of bibliographic databases have been described along with their basic characteristics. Some examples of bibliographic databases have been furnished by way of illustration. It has been stated that record formats like MARC, UNIMARC and MARC-21 are used for structuring bibliographic

databases and are important in the context of library automation and designing of bibliographic information systems. The Unit concludes with a mention of different 'views' of a 'database' and the three classical models of DBMS available for database design and architecture.

1.7 ANSWERS TO SELF CHECK EXERCISES

1) It is a new approach to designing and operating information systems. Database approach regards data as resource to be managed along with the more generally recognised resources of an organisation (such as staff, finance, equipment and premises) so as to be available to a variety of applications and users. An integrated database is intended to provide a consistent view of the institution's data for all user departments.

   The database approach is not dependent on any particular structure of data; many of these concepts can be applied to files as well. The database approach cannot be practically implemented in the absence of a computer.

   The attitudinal factors contributing to the database approach are:

   - Sharing valued data resources;
   - Releasing control of those resources to a common responsible authority; and
   - Cooperating in the maintenance of those shared data resources.

2) In the field of library and information science a database is defined as an organised and generally unlinked set of machine-readable bibliographic or source records.

   In order to understand the concept of bibliographic database, it is necessary to have some basic knowledge relating to files and records. A file is a collection of similar records with relationships defined between the records. A record is information contained in the database relating to *one document* or *item.* For example, in a catalogue database, a record may contain all information pertaining to a book. If, it is a source database, a record may contain the contents of a directory entry or an article of a journal. Each record in a file comprises a number of *fields.* Generally, there are two types of fields: *fixed length fields* and *variable length fields.* Fixed length fields are ideal to represent data elements of predetermined number of characters such as ISBNs and other type of information, which will be of the same length in each record. On the other hand, variable length fields would consist of different lengths in different records. Variable length fields pose a problem for the computer as it cannot recognise when one field ends and another starts. In other words, there should be a mechanism by which this problem can be solved. In fact, this is done by introducing the concept of flagging the *beginnings* and *ends* of the fields. Similar methodology is followed in the case of sub-fields. The method of dividing records into fields and sub-fields allows sub-sets of the database to be selected and retrieved as per the user's query pattern. It would also facilitate partitioning of the database for the purposes of management and adds precision for retrieval.

   Another important concept which needs to be understood in the context of a bibliographic database is *record format.* Attempts have been made to develop standard record formats. Such formats would be useful in library automation and would facilitate exchange of bibliographic records between different computer systems. Standard record formats like MARC- II, UNIMARC, and MARC-21 are commonly used in the construction of bibliographic databases. They embody agreements on the data elements of a bibliographic record.

3) The architecture of a database is commonly viewed in terms of three separate levels of descriptions; conceptual, external and internal.

The overall logical description is the conceptual level, which is commonly known as a schema. Subsets of schema that contain only the data needed for particular applications are called sub-schemas or user views. The sub-schemas provide a description at the external level. The description of physical storage structure used to store the database on a specific computer system is the internal description.

1.8 KEYWORDS

Bibliographic Database : A collection of bibliographic records in a machine-readable form. Typical databases include surrogates (i.e. author's name, title, abstract, etc) of library holdings or published journal articles.

Bibliographic Record : A collection of bibliographic data fields treated as one logical entity that describes a specific bibliographic item.

Database : Refers generally a machine-readable file of records but may be used more specifically to refer a shared collection of structured data managed by a set of special software.

Database Management Systems (DBMS) : The software used to manipulate and access data stored in database. A DBMS is generally designed to provide facilities for the maintenance of and access to structured data. The characteristics of data can be clearly defined with the help of a DBMS.

Field : Collection of data elements that together make up a unit of information. A field contains data elements such as the name of the author, title of the document, imprint, etc.

File : An organised collection of information and may consist of a set of logical records. It may be a named collection of computer records, usually with common attributes. A file may be stored and recalled as one unit by name.

Format : Structuring of fields in a record.

MARC Format : A bibliographic record format that has been devised for the MARC project.

Record : A collection of fields that form a logically related and discrete unit of information. For example, information regarding a library user constitutes a personal record.

1.9 REFERENCES AND FURTHER READING

Anderson, D. (1989). *Standard Practices in the Preparation of Bibliographic Records.* London: IFLA UBCIM.

Ashford, J.A. and Willet, P. (1989). *Text Retrieval and Document Databases.* Bromley: Chartwell Bratt.

Date, C.J. (1990). *Database: A Primer.* New York: Addison-Wesley Publishing.

Fidel, Raya (1987). *Database Design for Information Retrieval: A Conceptual Approach.* New York: John Wiley.

Gredley, E. and Hopkinson, A. (1990). *Exchanging Bibliographic Data: MARC and Other International Formats.* Ottawa: Canadian Library Association.

House, William C., (ed.) (1977). *Interactive Decision Oriented Database Systems.* New York: Petrocelli.

Judge, Peter and Gerrie Brender, (ed.) (1986). *Small Scale Bibliographic Databases.* Sydney: Academic Press.

Olson, Jack (1994). Defining a New State of Databases. *Informatics.* 2(3), 41-45.

Oxborrow, E.A. (1991). *Databases and Database Systems: Concepts and Issues.* Bromley: Chartwell Bratt.

Rowley, J. (1998). *The Electronic Library.* London: Library Association.

William, John (1992). *Database Design and Construction: An Open Learning Course for Students and Information Managers.* London: Library Association.

UNIT 2 DATA STRUCTURES, FILE ORGANISATION AND PHYSICAL DATABASE DESIGN

Structure

2.0 OBJECTIVES

After reading this Unit, you will be able to:

- understand the basic concepts related to data structures and file organisation;
- comprehend physical storage structures of data and file organisation techniques; and
- gain an insight into the role the data structures and file organisation play in the overall performance and access efficiency in a database.

2.1 INTRODUCTION

Data structures and file organisation refer to the methods of organising the data in a database. They primarily deal with physical storage of data, which assumes significance in retrieving, storing and re-organising data in a database. Data structures include linked

lists, inverted lists, B-trees and hash tables, among others. Data structures can be used to build data files (a data file or a file is a collection of many similar records) and file organisation determines access methods for the file.

File organisation (or file structure) is a combination of representations for data in files and of operations for accessing the data. A file structure allows applications to read, write, and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order. Data structures and file organisation define the physical design of a database and are critical to its performance.

2.2 DEFINITIONS AND BASIC CONCEPTS

Some of the basic concepts related to data structures and file organisation which are essential to provide the necessary background for their understanding are given in the following paragraphs:

2.2.1 Why Data Structures

The key factor in designing data structures and file organisation is the relatively slow speed of hard disks and large amount of time that is required to get information from a disk. All the data structures and file organisation designs focus on minimising disk accesses and maximising the likelihood that the information the user will want is already in the memory. The constraint related to disk access is generally referred to as I/O bottleneck. Accessing information using multiple trips to the disk greatly slows down the access time. Ideally, we should get the information we need with one access to the disk or with as few accesses as possible.

Tracking the developments of data structures and file organisation over the years, one finds that early work on files presumed that the files were on tape and access sequential. The cost of sequential access grew in direct proportion to the size of the file. As files grew intolerably large for unaided sequential access, indexes were added to the files. The indexes made it possible to keep a list of keys and pointers in a small file that could be searched more quickly. With the keys and pointers the user had direct access to the large, primary file. However, as the indexes grew, they too became difficult to manage, especially for dynamic files in which the set of keys changes. Then, in the early 1960s the idea of applying tree structures emerged which was refined over the years to provide a solution in the form of the B-tree. Furthermore, hashing technique was developed to provide fast access to files.

2.2.2 Memory Hierarchy

Computer storage media form a memory hierarchy that includes two main categories of storage:

- **Primary storage:** Pertains to storage media used by Central Processing Unit (CPU) i.e., the main memory and also the cache memory. The primary storage memory also called RAM (Random Access Memory) provides fast access to data and is volatile i.e., loses its content in case of a power outage.

- **Secondary storage:** Includes magnetic disks, optical disks and tapes. Secondary storage memory provides slower access to data than RAM.

  The memory hierarchy is represented in Fig. 2.1. As one moves down the hierarchy from cache memory, access speed and cost decrease.

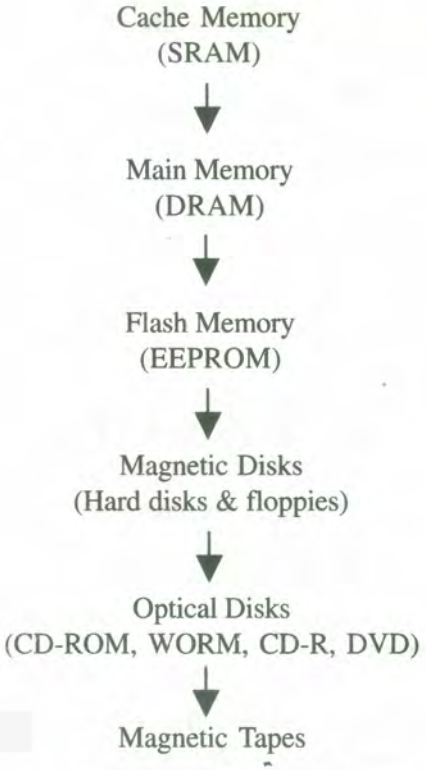


Fig. 2.1: Memory Hierarchy

Static RAM (SRAM) which is cache memory is used by CPU to speed up execution of programmes while Dynamic RAM (DRAM) provides the main work area for CPU. Flash memory which is non-volatile and called EEPROM (Electrically Erasable Programmable Read-Only-Memory) has access speed and performance between DRAM and magnetic disks.

CD-ROM (Compact Disk Read-Only-Memory) disks store data optically and are read by a laser. WORM (Write-Once-Read-Memory) disks are used for archiving data and allow data to be written once and read any number of times. DVD (Digital Video Disk) – a type of optical disk allows storage of four to fifteen gigabytes of data per disk. Magnetic tapes are used for archiving and back-up storage and are becoming popular as tertiary storage to hold terabytes of data. Juke boxes (optical and tape) are employed to use arrays of CD-ROMs and tapes.

2.2.3 RAID Technology

A major advance in secondary storage technology is represented by RAID (Redundant Array of Inexpensive/Independent Disks) technology. The RAID idea has been developed into an elaborate set of alternative RAID architectures (RAID levels 0 through 6).

The main goal of RAID is to even out the widely different rates of performance improvement of disks against those in memory and microprocessors. While RAM capacities have quadrupled every two to three years, disk access times are improving at less than 10 percent per year, and disk transfer rates are improving at roughly 20 percent per year. Though disk capacities are improving at a fast rate, the speed and access time improvements are of much smaller magnitude.

The problem of speed and access time is overcome by using a large array of small independent disks acting as a single high-performance logical disk. A concept called data striping is used, which utilises parallelism to improve disk performance. Disk striping exemplifies an important concept that we see more and more in system configuration—parallelism. Whenever there is a bottleneck at some point in the system, consider duplicating the source of the bottleneck and configure the system so that several of them operate in parallel. Data striping distributes data transparently over multiple disks to make them appear as a single large, fast disk. Striping improves overall I/O performance by allowing

multiple I/Os to be serviced in parallel, thus providing high overall transfer rates. Data striping also accomplishes load balancing among disks.

It should be noted that data can be read or written only one block at a time, so a typical transfer contains 512 bytes (block size = 512 bytes). Data striping can be applied at a finer granularity by breaking up a byte of data into bits and spreading the bits to multiple disks. Using bit-level data striping with 8-bit bytes, eight physical disks may be considered as one logical disk with an eight fold increase in data transfer rate. Each disk participates in each I/O request and the total data read per request is eight times. Data striping may also be done at block level which distributes blocks of a file across disks.

In addition to improving performance, RAID is also used to improve reliability by storing redundant information on disks. One technique for introducing redundancy is called mirroring. Data is written redundantly to two identical physical disks that are treated as one logical disk. If a disk fails, the other is used until the first is repaired.

Thus, RAID technology has contributed significantly in improving the performance and reliability of data storage on disks.

2.2.4 Indexes

An index is a file in which each entry (record) consists of a data value together with one or more pointers (physical storage addresses). The data value is a value for some field of the indexed file (the indexed field) and pointers identify records in the indexed file having that value for that field. The concept of indexing is closely linked with the operation of searching.

An index (sometimes also referred as a list) can be used in two ways. First, it can be used for sequential access to the indexed file, i.e., access according to the values of the indexed field by imposing an ordering of the indexed file. Second, it can also be used for direct access to individual records in the indexed file on the basis of a given value for that same field. In general, indexing speeds up retrieval but may slow down update.

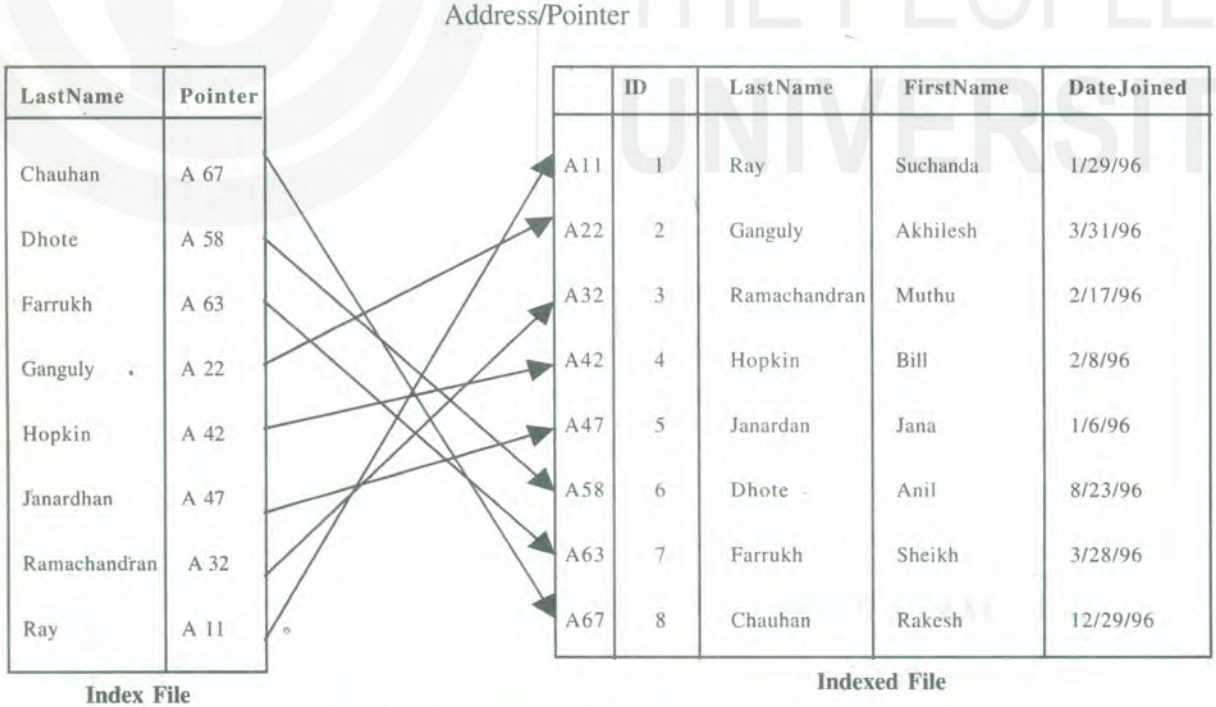


Fig. 2.2: An Illustration of Indexes

2.2.5 Binary Search

Binary searching is a technique used to substantially lessen the time required to search the indexes of lengthy inverted lists (see 2.3.2). In this technique, the value sought is first compared to the value in the middle of the list. This indicates whether the value sought is

in the top or bottom half of the list. The value sought is then compared with the middle entry of the appropriate half. This indicates which fourth the value is in. Then the value is compared to the middle of the fourth, and so on until the desired value is found. Thus the binary search keeps splitting the data set in half until it finds the desired value.

An example of binary search is shown in Fig. 2.3. To find the entry for Janardan find the middle of the list (Gautam). Janardan is post Gautam so split the second half in half (Kamla). Keep splitting the remainder in half until Janardan is found.

Fig. 2.3: An Example of Binary Search

Self Check Exercise

1) What is the role of indexes in database search?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

2.3 DATA STRUCTURES

The term Data Structure refers to the manner in which relationships between data elements are represented in the computer system. Organisation of indexes, representation of stored fields, physical sequence of stored records, etc., are included in the purview of data structures. Thus, an understanding of data structures is important in gaining an understanding of database management systems.

There are three major types of data structures : linked lists (indexes), inverted lists (indexes) and B-trees. These data structures have been explained in the following paragraphs:

2.3.1 Linked Lists

A simple linked list is a chain of pointers embedded in records. It indicates either a record sequence for an attribute other than the primary key or all the records with a common property. With a linked list, any data element can be stored separately. A pointer is then used to link to the next data item.

Fig. 2.4 illustrates the basic concept of linked lists. In this example each row of data is stored separately. Then an index is created on the field (key) Last Name. However, each element of the index is stored separately. An index element consists of three parts: the key value, a pointer to the rest of the data of that row, and a pointer to the next index element. To retrieve data sequentially, start at the first element (Chauhan) and follow the link (pointer) to the next element (Dhote). Each element of the index is found by following the link to the next element. The data pointer in each index element provides the entire data row for that key value. The strength of a linked list lies in its ability to easily and rapidly insert and delete data.

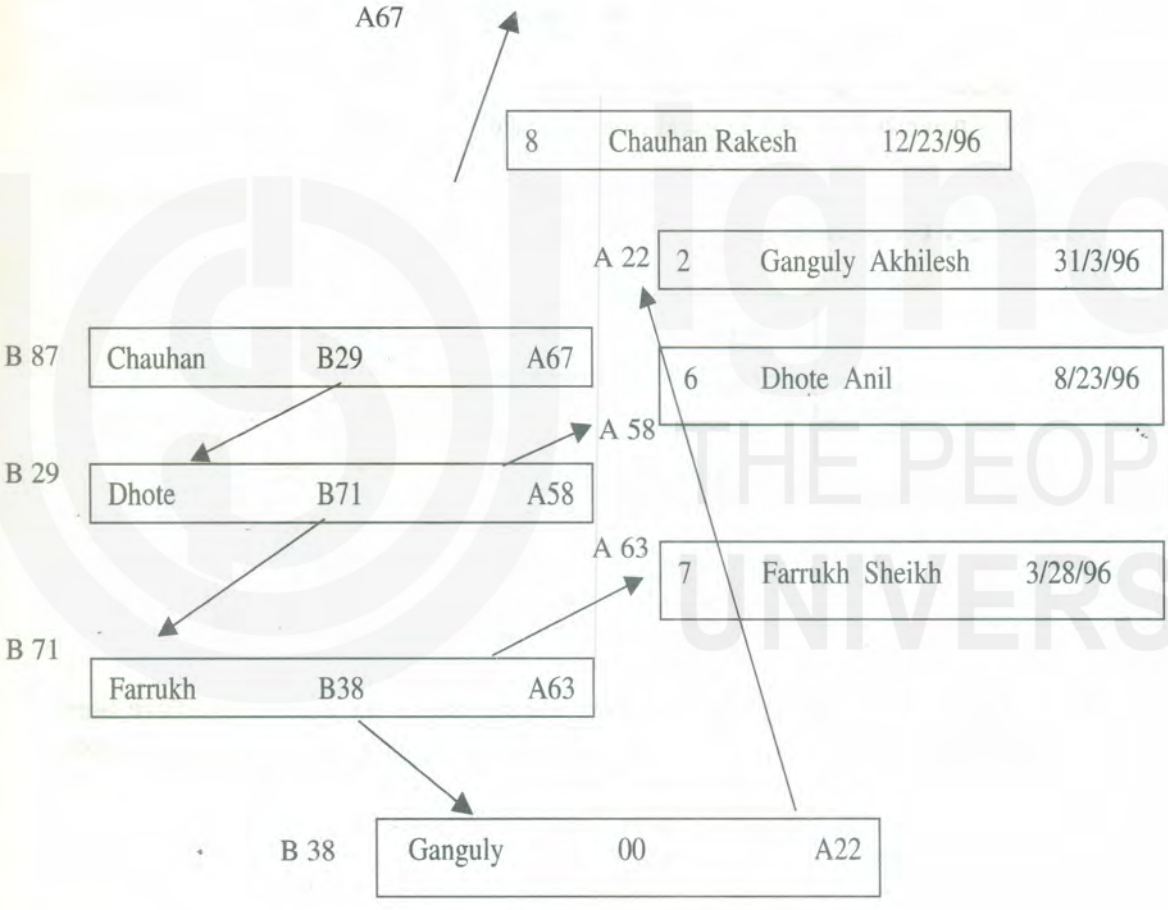


Fig. 2.4: An Illustration of Linked List

2.3.2 Inverted Lists

Inverted lists may be viewed simply as index tables of pointers stored separately from the data records rather than embedded in pointer fields in the stored records themselves.

Distinction should be made between nondense and dense lists. In case of a nondense list only a few of the records in the file are part of the list while a dense list is one with a pointer for most or all of the records in the file.

Processing for unique secondary keys (those having 1:1 association with primary keys) is somewhat different than those with 1:M associations with primary keys. In the former case, dense indexes are generated while the later gives nondense indexes.

Examples of inverted lists are given below:

List 1

Company	Area	Primary Key
Digital	Computer	1245
Ford	Auto	1175
GM	Auto	1323
Intel	Computer	1231
Lockheed	Aerospace	1152

Index File

List 2

Company Symbol	Primary Key
DEC	1245
F	1175
GM	1323
INTL	1231
L	1152

Indexed File

Fig. 2.5: Dense Inverted Lists

The above lists are dense since there is one-to-one relationship between both company name and primary key and company symbol and primary key.

Fig. 2.6 gives an example of non-dense inverted list for area (relationship between area and primary key is one-to-many).

Area	Primary Key
Aerospace	1152
Auto	1175, 1323
Computer	1231, 1245

Fig. 2.6: Nondense Inverted List

The lists are said to be inverted because company names (or area names) have been alphabetized and the corresponding primary keys have been "inverted" or rearranged accordingly.

2.3.3 B-Trees

B-trees are a form of data structure based on hierarchies. Some authors claim that the letter "B" stands for Bayer, the originator while others say it stands for "balanced". B-trees are balanced in the sense that all the terminal (bottom) nodes have the same path length to the root (top). Algorithms have been developed for efficiently searching and maintaining B-tree indexes, which have become quite popular for representing both primary and secondary indexes. B-trees provide both sequential and indexed access and are quite flexible.

The height of a B-tree is the number of levels in the hierarchy. Each node on the tree contains an index element which has a key value, a pointer to the rest of the data and two link pointers (see Fig.2.7) One link (to the left) points to the elements (nodes) that have lower values while the other link (to the right) points to elements that have a value greater than or equal to the value in the node. The root is the highest node on the tree. The bottom nodes are called leaves because they are at the end of the tree branches.

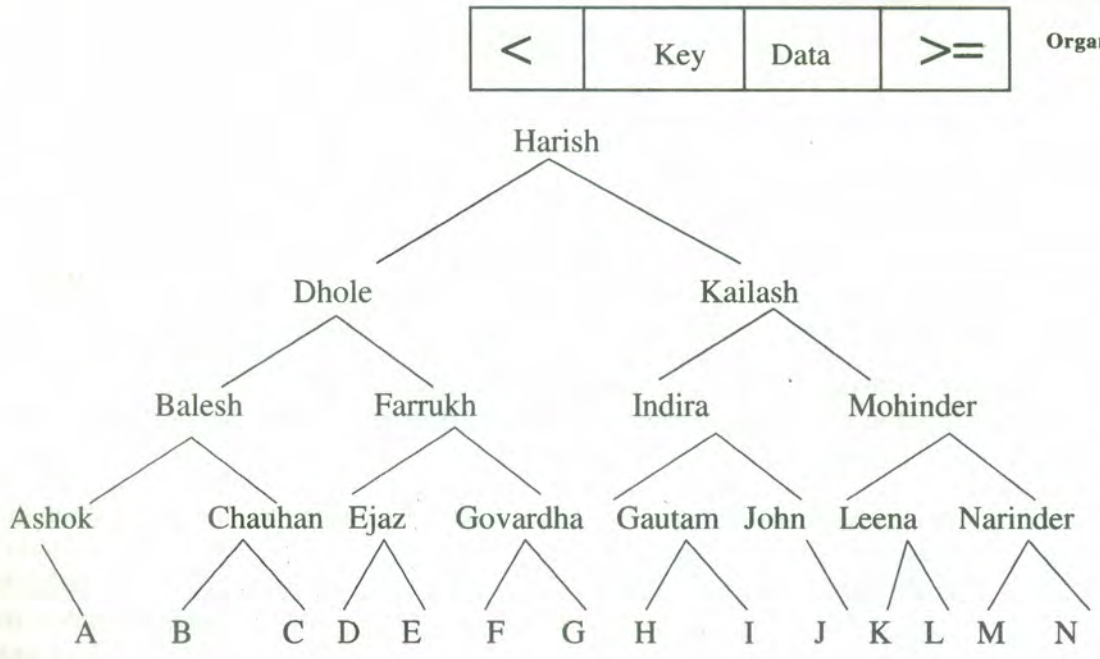


Fig. 2.7: An Illustration of a B-tree

A B-tree is called unbalanced if the terminal nodes (leaves) are not all at the same level i.e., if different terminal nodes are at different depths below the top node.

B-trees provide excellent access performance but do not allow a file to be accessed sequentially with efficiency. This problem is overcome by adding a linked list structure at the bottom level of the B-tree. The combination of a B-tree and a sequential linked list is called a B^+ tree.

Self Check Exercise

2) Give examples of commercially available database management systems, which use B-trees/ B^+ trees.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

2.4 FILES AND THEIR ORGANISATIONS

2.4.1 File Storage Concepts

A file is a sequence of records. File organisation refers to physical layout or structure of record occurrences in a file. File organisation determines the way records are stored and accessed.

In many cases, all records in a file are of the same record type. If every record in the file has exactly the same size (in bytes), the file is said to be made of fixed-length records. If

different records in the file have different sizes, the file is said to be made up variable-length records. A file may have variable-length records for several reasons:

i) The file records are of the same record type, but one or more fields are of varying sizes (variable-length fields).

ii) The file records are of the same record type but one or more fields may have multiple values for individual records. Such a field is called repeating field and a group of values for the field is often called a repeating group.

iii) The file records are of the same record type, but one or more fields are optional.

iv) The file has records of different record types and hence of varying size (mixed file). This would occur if related records of different types were clustered (placed together) on disk blocks.

The records of a file must be allocated to disk blocks because a block is a unit of data transfer between disk and memory. The division of a track (on storage medium) into equal sized disk blocks is set by the operating system during disk formatting. The hardware address of a block comprises a surface number, track number and block number. Buffer – a contiguous reserved area in main storage that holds one block has also an address. For a read command, the block from disk is copied into the buffer, whereas for a write command the contents of the buffer are copied into the disk block. Sometimes several contiguous blocks, called a cluster, may be transferred as a unit. In such cases buffer size is adjusted to cluster size.

When the block size is larger than the record size each block will contain numerous records, while there can be files with large records that cannot fit in one block. In the latter case the records can span more than one block.

Here it is worthwhile to note the difference between the terms File Organisation and Access Method. A file organisation refers to the organisation of the data of a file into records, blocks and access structures; this includes the way the records and blocks are placed on the storage medium and interlinked. An access method on the other hand, provides a group of operations – such as find, read, modify, delete, etc, — that can be applied to a file. In general, it is possible to apply several access methods to a file organisation. Some access methods, though, can be applied only to files organised in certain ways. For example, we cannot apply an indexed access method to a file without an index.

2.4.2 Sequential Access Method (SAM)

In sequential files, records are stored in a predefined order. Their occurrences in a sequential file are usually sorted on the primary key and physically arranged on the storage medium in order by primary key. If only sequential access is required (which is rarely the case), sequential media (magnetic tapes) are suitable and probably the most cost-effective way of processing such files. Direct access devices such as disks may be, but are not necessarily, referenced sequentially. Some types of processing are best done through sequential access, even when direct access devices are used.

Sequential access is fast and efficient while dealing with large volumes of data that need to be processed periodically. However, it requires that all new transactions be sorted into proper sequence for sequential access processing. Also, most of the database or file may have to be searched to locate, store, or modify even a small number of data records. Thus, this method is too slow to handle applications requiring immediate updating or responses.

Sequential files are generally used for backup or transporting data to a different system. A sequential ASCII file is a popular export/import format that most database systems support.

2.4.3 Indexed Sequential Access Method (ISAM)

In indexed sequential files, record occurrences are sorted and stored in order by primary key on a direct access storage device. In addition, a separate table (or file) called an **index** is maintained on primary key values to give the physical address of each record occurrence. This approach gives (almost) direct access to record occurrences via the index table and sequential access via the way in which the records are laid out on the storage medium.

The physical address of a record given by the index file is also called a **pointer**. The pointer or address can take many forms depending on the operating system and the database one is using.

Nowadays, systems use virtual addresses instead of physical addresses. A virtual address could be based on imaginary disk drive layout. The database refers to a base set of tracks and cylinders. The computer then maps these values into actual storage locations. This arrangement is the basis for an approach known as the virtual sequential access method (VSAM). Another common approach is to define a location in terms of its distance from the start of a file (relative address). Virtual or relative addresses are always better than the physical address because of their portability.

In case a few records need to be processed quickly, the index is used to directly access the records needed. However, when large numbers of records must be processed periodically, the sequential organisation provided by this method is used.

An illustration of access using index file is given in Fig. 2.8.

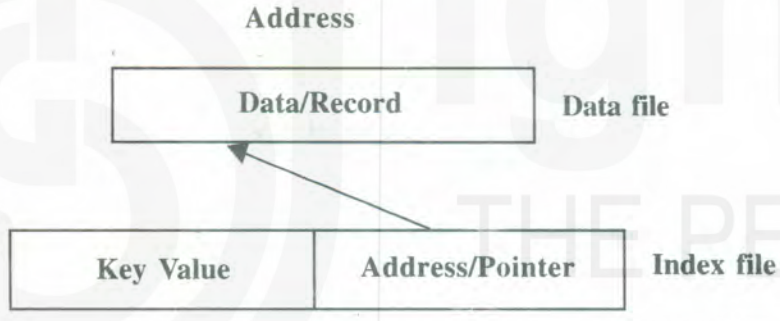


Fig. 2.8: Data Access Using Index File

2.4.4 Direct Access Method (DAM)

When using direct access method, the record occurrences in a file do not have to be arranged in any particular sequence on storage media. However, the computer must keep track of the storage location of each record using a variety of direct organisation methods so that data is retrieved when needed. New transactions data do not have to be sorted, and processing that requires immediate responses or updating is easily handled.

In direct access method an algorithm is used to compute the address of a record. The primary key value is the input to the algorithm and the block address of the record is the output.

To implement the approach, a portion of the storage space is reserved for the file. This space should be large enough to hold the file plus some allowance for growth. Then the algorithm that generates the appropriate address for a given primary key is devised. The algorithm is commonly called hashing algorithm. The process of converting primary key values into addresses is called key-to-address transformation.

More than one logical record usually fits into a block, so we may think of the reserved storage area as being broken into record slots sequentially numbered from 1 to n. These sequential numbers are called relative pointers or relative addresses, because they indicate the position of the record relative to the beginning of the file.

The objective of the hashing algorithm is to generate relative addresses that disperse the records throughout the reserved storage space in a random but uniform manner. The records can be retrieved very rapidly because the address is computed rather than found through table look-up via indexes stored on a disk file.

A collision is said to occur if more than one record maps to the same block. Because one block usually holds several records, collisions are only a problem when the number of records mapping to a block exceeds the block's capacity. To account for this event, most direct access methods support overflow area for collisions, which is searched sequentially.

The hashed key approach is extremely fast since the key's value is immediately converted into a storage location, and data can be retrieved in one pass to the disk. An illustration of direct access method using hashed key is given in Fig. 2.9.

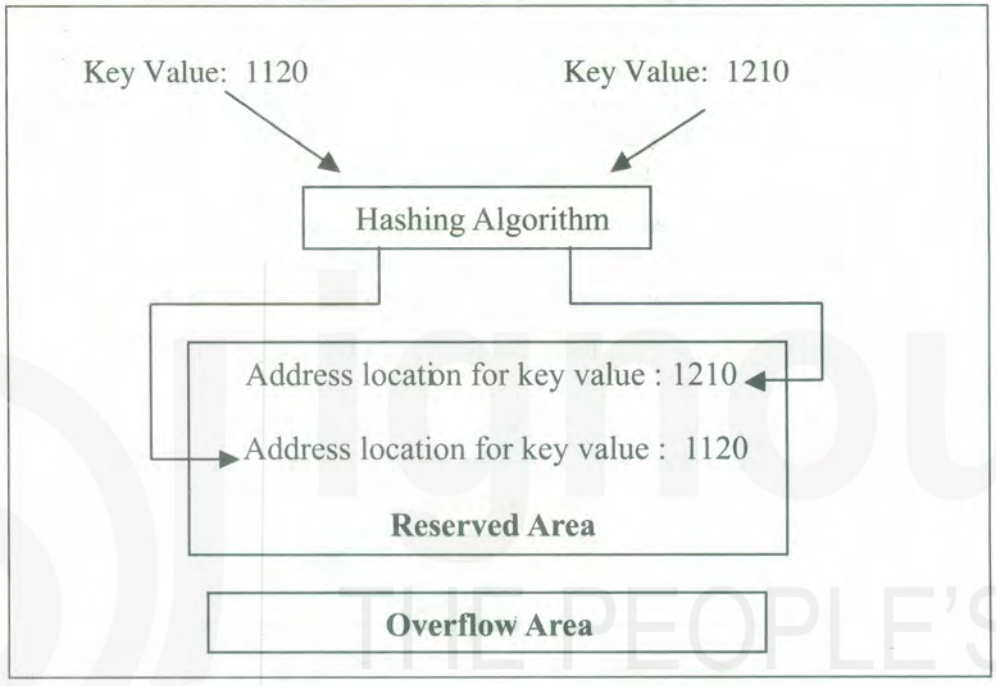


Fig. 2.9: Direct Access Using Hashed-Key Approach

2.5 PHYSICAL DATABASE DESIGN

Data structures and file organisation play an important role in physical database design (please refer to Unit 3 on Database Management Systems for database architecture and database design issues). The goal of the physical design is to come up with appropriate structuring of data in storage so as to ensure good performance of the database. It is not possible to make meaningful physical design decisions and performance analyses until we know the queries, transactions and applications that are expected to run on the database.

The following aspects influence the physical database design:

1) **Analysing the database queries and transactions**:

   For each query we should specify:

   a) The files that will be accessed by the query.

   b) The attributes on which any selection conditions for the query are specified.

   c) The attributes on which any join conditions or conditions to link multiple tables for the query are specified.

   d) The attributes whose values will be retrieved by the query.

The attributes at b) and c) are candidates for definition of access structures. For each update transaction we should specify:

a) The files that will be updated.

b) The type of operation on each file (insert, update or delete).

c) The attributes on which the selection conditions for a delete or update are specified.

d) The attributes whose value will be changed by the update operation.

Here the attributes at c) are candidates for access structures and attributes at d) are candidates for avoiding an access structure since modifying them will require updating the access structures.

2) **Analysing the expected frequency of queries and transactions:** This yields the expected frequency of using each attribute in each file as a selection attribute or a join attribute, over all the queries and transactions.

3) **Analysing the time constraint of queries and transactions:** Some queries and transactions may have stringent performance constraints with respect to response time. The selection attributes used by queries and transactions with such time constraints become higher priority candidates for primary access structures.

4) **Analysing the expected frequencies of update operations :** A minimum number of access paths should be specified for a file that is updated frequently, because updating the access paths themselves slows the update operations.

Based on the preceding information one can address the physical database design decisions about indexing. The attributes whose values are required in equality or range conditions (selection operation) and those that are keys or that participate in join conditions (join operation) require access paths. The performance of queries largely depends upon what indexes or hashing schemes exist to expedite the processing of selections and joins. On the other hand, during insert, delete, or update operations, existence of indexes adds to the overhead.

The following points may be kept in view while taking decisions for indexing:

i) The attribute, which is to be indexed, must be a key or there must be some query that uses that attribute either in selection condition (equality or range of values) or in a join.

ii) An index can be made on one or multiple attributes. If multiple attributes from one relation are involved together in several queries, a multiattribute index is warranted.

iii) Clustering index (index created on a non-key field i.e., if numerous records in a file can have the same value for the field) can be greatly useful in range queries. If several attributes require range queries relative benefits must be evaluated before deciding which attributes to cluster on. At most one index per table can be a primary or clustering index.

iv) RDBMSs generally use B^+ trees for indexing. ISAM and hash indexes are also provided in some systems. B^+ trees support both equality and range queries on the attribute used as the search key. Hash indexes work well with equality conditions, particularly during joins.

2.6 SUMMARY

This Unit covers some of the key issues related to data structures, file organisation and physical database design and provides essential background to facilitate their understanding. The role of data structures and file organisation on the performance of access methods has been explained. RAID technology, binary search and indexes have been discussed to elucidate how access speed can be improved. Typical data structures (linked lists, inverted lists and B-trees) and file organisation techniques (SAM, ISAM and DAM) have been

dealt with. The factors, which influence the physical database design and decisions on access structures, have been explained.

This Unit lays the foundation for understanding the complex key concepts related to the topic.

2.7 ANSWERS TO SELF CHECK EXERCISES

1) Existence of appropriate indexes is critical in data retrieval. In systems where query response time is a major consideration, indexes are used to speed up access. However, presence of indexes tends to slow down updating process.

2) A number of commercially available DBMS have B-tree / B$^+$tree index creation features built into the system i.e., the system automatically generates B$^+$tree indexes for speeding up and optimising queries. ORACLE and SYBASE are examples of relational database management systems, which support B-tree/ B$^+$tree indexes.

2.8 KEYWORDS

Cache Memory : A high speed temporary storage in the CPU for storing parts of a program or data during processing.

Binary Search : A search technique for sorted data.

B-tree : An indexed data storage method that is efficient for a wide range of data access tasks.

B$^+$-tree : A variation on the B-tree structure that provides sequential access to the data as well as fast-indexed access.

Hashing : An access mechanism that transforms the search key into a storage address, thereby providing very fast access to stored data.

Index : A sorted list of key values from the original table along with a pointer to the rest of the data in each row.

Pointer : A logical or physical address of a piece of data .

RAID : Redundant Array of Independent Disks. A disk drive system that consists of multiple drives with independent controllers. The goal is to split the data to provide faster access and automatic duplication for error recovery.

Sequential Access : Access that takes records in order, looking at the first, then the next, and so on.

2.9 REFERENCES AND FURTHER READING

Courtney, James F. and Paradice, David, B. (1988). *Database Systems for Management.* Toronto: Times Mirror/Mosby College Publishing.

Date, C.J. (1989). *Introduction to Database Systems*. New Delhi: Narosa Publishing House.

Elmasri, Ramaz and Navathe, Shaukan, B. (2000). *Fundamentals of Database Systems.* Asia: Pearson Education.

Folk,, Michael, J. [et.al.] (2004). *File Structures: An Object-oriented Approach with C^{++}*. New Delhi: Pearson Education.

Gerald, V. Post (2000). *Database Management Systems.* New Delhi: Tata McGraw-Hill.

O'Brien, James A. (1997). *Introduction to Information Systems*. Irwin: The McGraw-Hill Company.

UNIT 3 DATABASE MANAGEMENT SYSTEMS

Structure

3.0 OBJECTIVES

After reading this Unit, you will be able to:

- understand the evolutionary pattern of development of database approach;
- comprehend the need for a database management system (DBMS), its primary objectives, database architecture and basic issues in the complex and expanding environment of database management;
- distinguish between various data models and know the characteristics of each;
- become conversant with relational database technology; and
- gauge emerging trends in data management.

3.1 INTRODUCTION

Database systems have permeated all spheres of life. This Unit provides core information in understanding database management systems. Starting with the need for a database approach, the three-level database architecture has been explained. Modeling concepts and the role E-R model plays in conceptual database design has been discussed. Classical data models viz., hierarchical model, network model and relational model have been explained with illustrations. Considering that the Relational Database Management Systems (RDBMS) are still widely in use, relational database technology has been dealt in details. The concepts of dependencies and normalisation have been elucidated with examples. A step-by-step approach for designing databases has been given. The Unit also dwells on some of the database systems in specific application areas.

3.2 DEFINITIONS AND BASIC CONCEPTS

Some commonly used database terms are defined in the following paragraphs to help you understand the data architecture concepts.

3.2.1 Data and Information

Data is the raw material from which information is derived as the end product. Data represents a set of characters that have no meaning on their own, i.e., it consists of just symbols. On processing, meaning is attached to data, which transforms into information.

To illustrate the difference between these two terms let us consider an example. The digits 050643 as such have no meaning. But if we are told that the first two digits represented a month, the next two digits, a day of the month and the last two, a year, then the set 050643 may represent the date of birth of a person. Processed in another manner the same digits written as 643050 may represent the telephone number of an individual.

3.2.2 Database and Database Management System (DBMS)

Database

A database is a collection of logically related data arranged in a structured form designed to meet the information requirements of multiple users. It may also be defined as a collection of non-redundant operational data sharable between different application systems.

Database Management System

It is a collection of software that is used to store, delete, modify and retrieve data that is stored in a database. DBMS acts as an interface between the user and the data (Fig. 3.1).

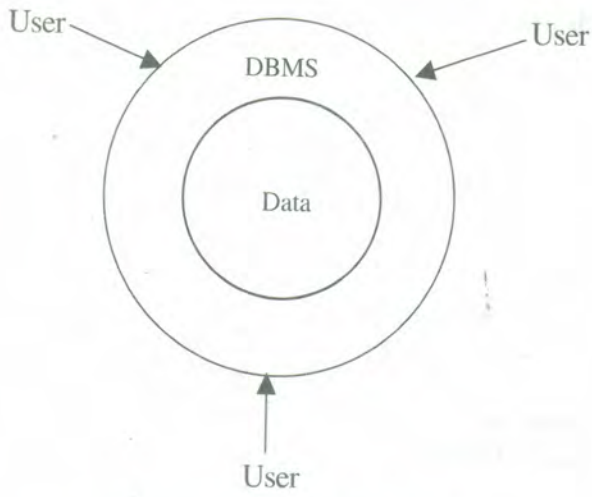


Fig. 3.1: Data, DBMS and Users

3.2.3 Data Hierarchy

Hierarchy in the organisation of data in descending order of complexity is represented in Fig. 3.2.

Fig. 3.2: Data Hierarchy

From this hierarchy it is clear that a database is made up of files. Files are composed of records and each record consists of fields or data items. Each field is composed of characters, which are made up of bytes. And lastly, bytes decompose into bits.

3.2.4 Data Integrity

The degree of correctness, timeliness, and relevance of data stored in a database indicates data integrity. Data integrity can be ensured by certain checks and validation procedures carried out whenever an update operation is attempted and also by elimination of data redundancy. Some database management systems have features, which support data validation. For example, in ORACLE (a relational database management system) triggers can be used for this purpose.

3.2.5 Data Independence

Data independence is a property by which data files are insulated from application programs that use those files. The close link between the data and the access program is weakened and the database made more flexible to user requirements. With data independence, a programmer can write programs in FORTRAN, PASCAL, or any other language of his choice without bothering what language programs originally created the files that he wants to access.

Data independence can be logical, physical or geographical. By logical or physical data independence we mean the ability to change the logical or physical structures of data without requiring any changes in the application programs which manipulate that data. Geographical data independence, a characteristic of distributed database management systems, makes location of data transparent to the users.

Data independence is an important concept which will be considered in more details while discussing the architecture of database management systems.

Self Check Exercise

1) What is data independence and what role does it play in database systems?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.3 OBJECTIVES OF DATABASE MANAGEMENT SYSTEMS (DBMS)

3.3.1 The Need for DBMS

Let us consider the scenario of data processing that existed before the advent of database management systems. In the file-oriented system, which was used at that time, a master file (the file which contains all the up-to-date data on a subject) is created using a programming language. Access techniques based on the requisite queries on the data are embedded in the file at the time of its creation. Any change in the master file i.e., addition of a new field or change in the structure of an existing field has to be implemented by creating the file all over again along with the modified access techniques.

To illustrate the limitations of data management using master files, let us consider the example of a database of an educational institution running professional courses. Let us assume that the database consists of three master files of student, faculty and course records. The student master file has been created using FORTRAN and has such fields as student identification number, name, address, gender, course, high-school grade and examinations cleared.

The faculty master file uses COBOL and consists of fields like faculty identification number, name, gender, department, salary, qualifications and teaching hours. The course master file is based on PASCAL and covers such data as course identification number, course title, class number, section number and students attending the course.

Now, suppose there is a query to provide the names of all the female students being taught by female faculty members. This query cannot be answered by the available master files despite the fact that the data needed for the query exists in the database. The difficulty lies in the fact that the needed data is available in two master files created in different programming languages and having their own access techniques. To answer the query a new master file with data items derived from the student and faculty master files will have to be created and new programmes for accessing the data written. This makes data retrieval cumbersome and time-consuming.

Take another situation when the accounts department of the institution in the example also wants to use the database and needs the student master file with additional fields like stipend paid, fees due, penalties charged, etc. To meet these requirements, another copy of the student master file with new fields is created. Similarly, there may be copies of faculty and course master files created to meet specific requirements. This results in duplication of data i.e., data redundancy. In such circumstances it becomes difficult to keep the master files identically updated i.e., propagate the updates in all the copies.

These limitation and drawbacks were at the core of development of database management systems.

3.3.2 The Goals of DBMS

The database management systems have the following goals :

i) To provide retrieval flexibility. It should be relatively easy to link data from different files.

ii) To facilitate reduction of data duplication and elimination of multiple copies of a master file. Data redundancy control helps in overcoming updating problems and promotes data integrity.

iii) To ensure high level of data independence. The data is hidden from the programming language, operating system and processing environment. It should be up to DBMS to convert the stored data into a form that could be used in whatever language the programmer desires to use.

3.4 EVOLUTION OF DBMS

The database management concepts have undergone tremendous changes over the years. The chronological evolution of database management systems is described below:

3 4.1 Chronology

i) **Primitive (first generation) Systems**

This stage is illustrated in Fig. 3.3a. Here, programming overheads are high as the user has the responsibility for all the I/O (Input/Output) programming as well as file management. Application programs (software that supports end user activity) have to be written to access the data stored in secondary storage devices (Direct Access Storage Devices). This phase represents the period of mid 1960s.

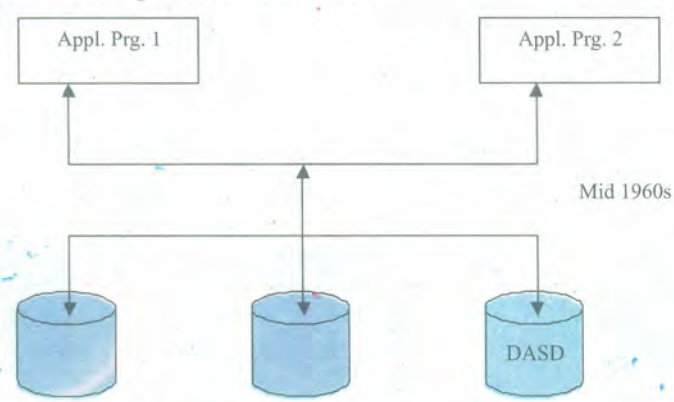


Fig. 3.3(a): First Generation Systems

ii) Second Generation Systems

Appearing in late 1960s, this generation is represented in Fig. 3.3b. In this case, operating system takes care of file management and other routines thereby relieving the programmer from this effort. Though physical data independence is achieved, logical data independence remains to be built into the system.

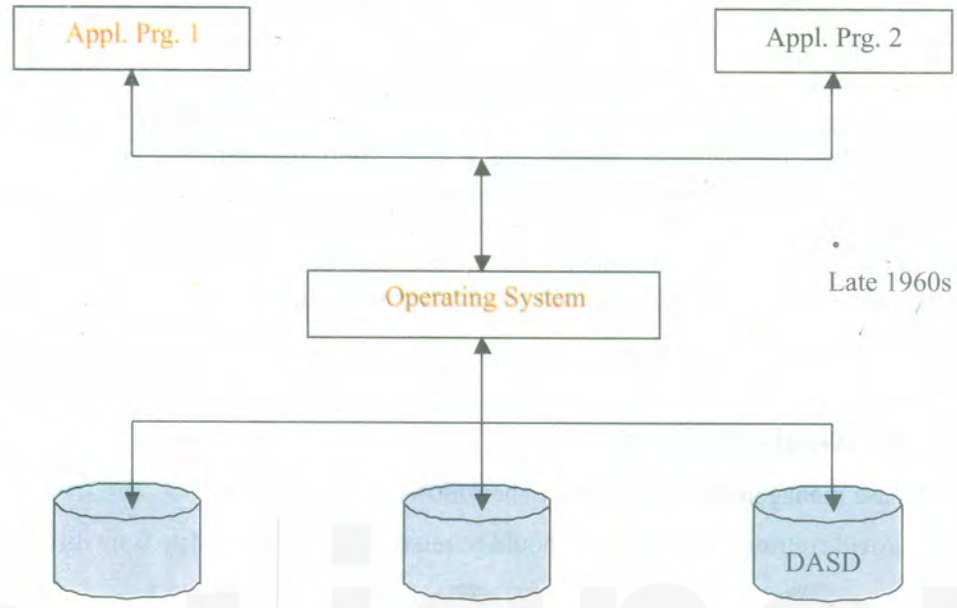


Fig. 3.3(b): Second Generation Systems

iii) Third Generation Systems

The third generation systems (Fig. 3.3c) evolved in early 1970s. Here, a layer of DBMS has been added over the operating system. The systems provide physical as well as logical data independence. The advantage it gives is that query processing can be attempted by offering higher levels of operation on the logical view of the data available from DBMS to the application programs.

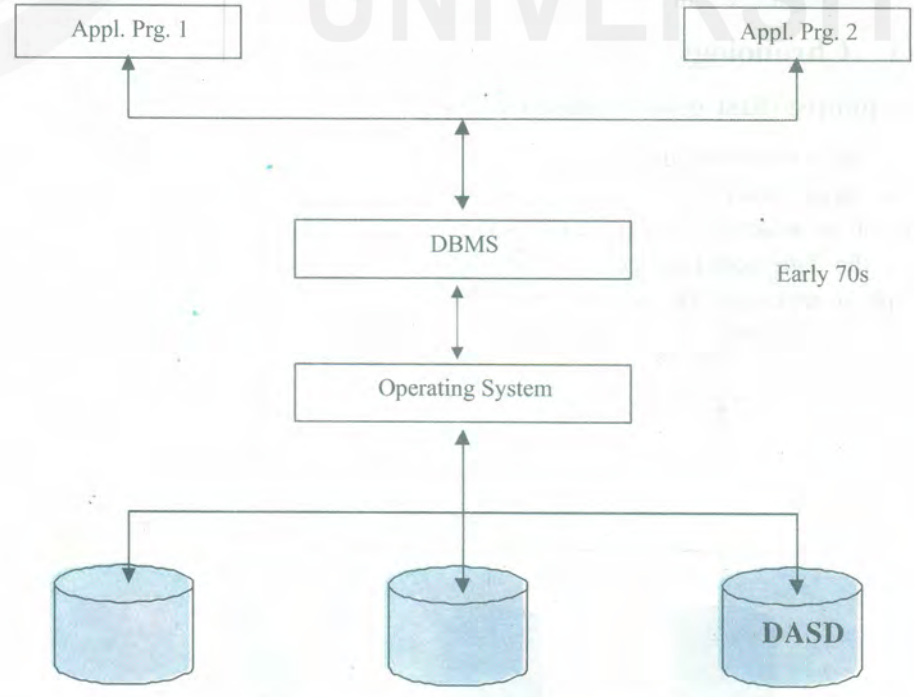


Fig. 3.3(c): Third Generation Systems

3.4.2 Functions and Components of a DBMS

Basically, there are only two operations that can be performed on data viz., retrieval and maintenance. Retrieval refers to reading data from files to serve the information requirements of a user and forms the most important function of a database management system. Maintenance concerns changing of data in stored files.

Data maintenance involves three operations: addition, deletion and modification which correspond to adding new records, deleting existing records and modify/updating values in the existing records.

A database management system has two essential components: data definition part and data manipulation part. Data definition part provides definition or description of database objects and is written using data definition language (DDL). This part creates logical structures of entities when a database is set up.

Data manipulation part refers to methods of manipulating data and is implemented using data manipulation languages (DML). There are four basic methods of data manipulations : programming language interface, query languages, report writers and system utilities.

Programming language interface (PLI) or host language interface provides access to the database through some type of programming language (PASCAL, C, COBOL, etc.). Query languages allow fast retrieval of data and some of them are considered fourth generation languages (4GL). The 4GLs are non-procedural languages, which implies that a user has to specify only what data is required and not how it should be retrieved.

The query languages can be grouped into two categories – command-driven query languages and screen-oriented query languages. In the first case the commands are specified in English-like text while in the second case the user enters commands through a fill-in-the blank mechanism. SQL (structured query language), a 4GL and a standard language for interfacing with relational DBMS, belongs to the first category while querying data through SQL forms is an example of the second category.

Report writers represent programs, which are used to derive information from a database and generate a report that presents this information in the desired fashion. And lastly, system utilities, are programs which allow a system manager to take back-up of databases, load data into a database, restore data in case of database crash and carry out other jobs related to database administration.

3.5 ARCHITECTURE OF A DBMS

Database management systems have a three-level architecture (see Fig. 3.4). Schema, level and view are used interchangeably to describe the architecture of a DBMS. The uppermost level in the architecture is the external level, which refers to the way the users view the data. The external level is also sometimes called subschema. A user would generally be interested only in some portion of the total database, which forms his external view. There may be several external views of the same database depending upon the user requirements. External schema can be used for implementing data security by restricting access to the database.

The second level is the conceptual schema, which represents the entire information content of the database. It gives global or integrated view of the database – the view of the database administrator. The conceptual schema is written using the conceptual DDL (Data Definition Language).

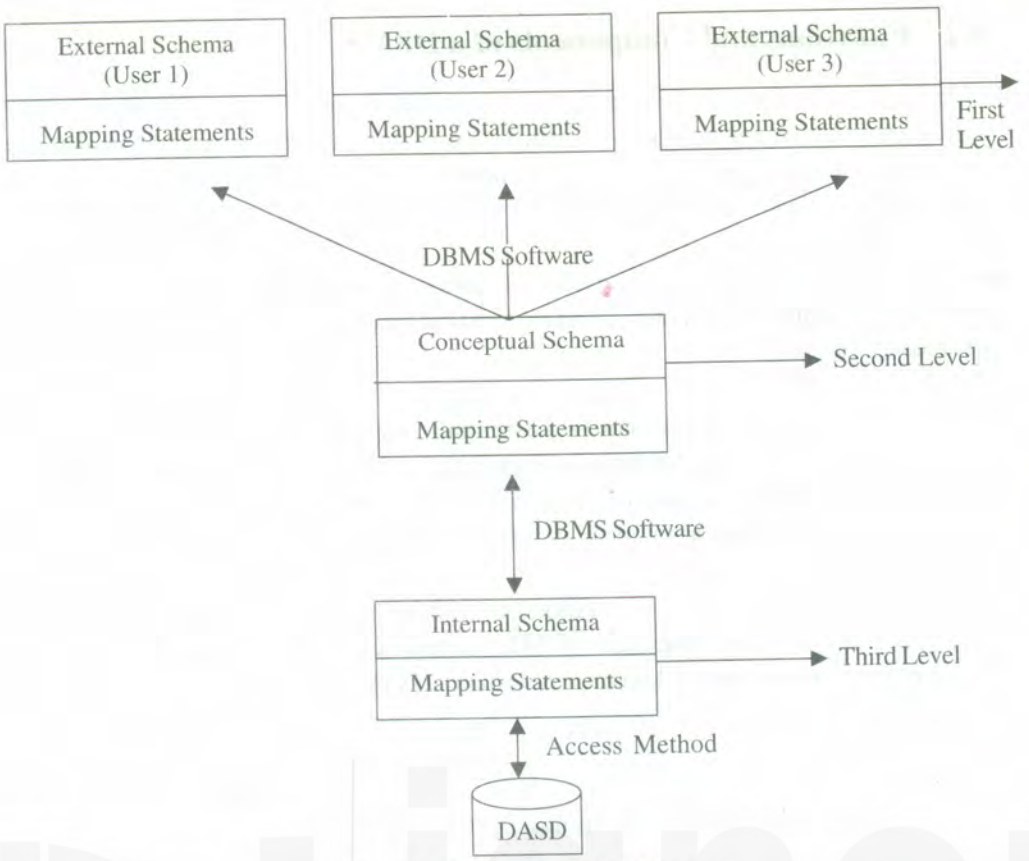


Fig 3.4: Architecture of a DBMS

Mapping statements (software programs) at the first level which establish correspondence between the external and conceptual schemas provide data independence which implies that one can define additional database objects or modify the structures of the existing ones without requiring any existing user to change his application programs. With logical data independence, a conceptual schema can grow and evolve with time without affecting the external schemas.

The third level of the architecture is the internal schema. Internal view describes how the data is actually stored and managed on the computer's secondary storage. It specifies what indexes exist, how stored fields are represented, physical sequence of stored records, etc. The internal schema is written using the internal DDL. Conceptual/internal mapping statements ensure physical data independence – that is the way the data is actually stored can be changed without requiring any change in the conceptual schema.

The mapping component between internal schema and secondary storage devices (direct access storage devices) is called access method. The access method consists of a set of routines whose function is to conceal all the device-dependent details from the DBMS and present the DBMS with the stored record view, i.e., the internal view. In many cases the underlying operating system performs this function.

Self Check Exercise

2) What is a schema and a sub-schema in database system? Briefly explain the purpose of the views in database architecture.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

3.6 DATA MODELING

Data modeling is a process by which the data requirements of an organisation or an application area are represented. Conceptual modeling is an important phase in designing a successful database application. The traditional approach is of using entity-relationship (ER) model for this purpose. Enhanced ER or EER model is used for modeling object-oriented databases.

3.6.1 Entity-Relationship Model

Entity-relationship model developed by Chen in 1976-77 serves as an excellent tool in the database design process. It provides graphic representation of entities, attributes and relationships.

Requirements analysis of the designed database helps in collecting the necessary information in the form of entities and attributes to be included in the database. Based on enterprise rules, the relationships between the entities get identified and the nature of use of the database determined. E-R model describes the conceptual schema and is considered as a blueprint of the database under design. After finalisation, E-R diagram (as entity-relationship model is generally called) is mapped into one of the selected database models (discussed later in the text) and the system-dependent procedure of database creation started.

An example of E-R diagram is illustrated in Fig. 3.5. It depicts a database on marketing of drugs from medicinal and aromatic plants. The plants or their parts serve as crude drugs, which are traded in the market. The standardising agency certifies the quality of drugs while certifying agency approves the drugs for export. Before supply to the customer the crude drugs are sometimes processed.

In an ER diagram rectangles represent entities, ellipses—attributes, diamonds—relationships, attributes with underscore—primary keys; attributes with double underscore—foreign keys and 1, n, m—relationship types.

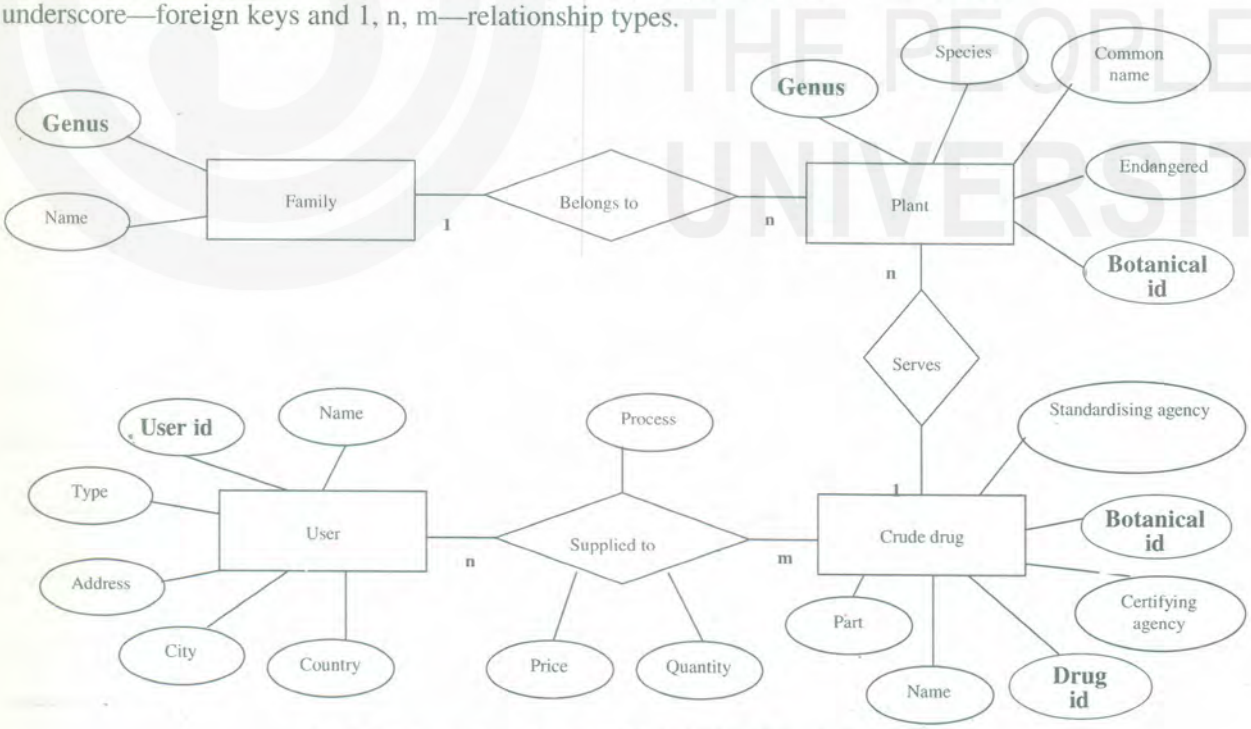


Fig. 3.5: Illustration of an E-R Diagram

3.6.2 Types of Relationships

A relationship is an association between two or more entities. Entities correspond to record types in a database, which are sets. Thus a relationship represents correspondence between the elements of n sets. Relationship over 2 sets is called binary relationship; relationship over 3 sets—ternary and over n sets—n-ary relationship.

Relationships can themselves be treated as entities and assigned attributes (see Fig. 3.5). Relationships can be grouped into the following types :

i) 1:1 (one-to-one)

ii) 1:n (one-to-many)

iii) n:1 (many-to-one)

iv) n:m (many-to-many)

Let us illustrate these relationship types one by one. In 1:1 relationship, one instance of an entity of a given type is associated with only one member of another type. Let there be a set of country names and a set of city names. Further, let us assume that each city in the set is a capital. The relationship between these two sets which can be called "capital" is 1:1 because for each country name there is only one city name and conversely, each city name corresponds to only one country name.

In 1:n relationship, one instance of a given type of entity is related to many instances of another type. Let there be a set of departments and a set of faculty members employed in the departments. Department-faculty relationship which can be called 'employed' is of 1:n type because each department employs several faculty members and each faculty member works only in one department.

Many-to-one (n: 1) relationship has the same semantics as 1:n. In the above example if we change the relationship to faculty-department (in place of department-faculty) we will have n: 1 relationship type.

Lastly n:m relationship is one in which many instances of an entity type are associated with many instances of another entity type. Consider a set of a faculty members teaching a set of students. Faculty-student relationship ("teaching") is an example of n:m type relationship because a faculty member can teach 'm' students and a student can be taught by 'n' faculty members.

Self Check Exercise

3) What is the significance of an entity-relationship (ER) diagram?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.7 DATA MODELS

The data models are methods by which data is structured to represent the real world and the manner in which the data is accessed. These models provide alternative ways of picturing the relationships and serve as frameworks for mapping the conceptual schema of a database. There are a number of data models in use today but the following three have been most widely implemented:

- Hierarchical
- Network
- Relational

Besides these three models, which are sometimes referred to as classical models, post-relational research has resulted in a new data model called object-oriented data model.

3.7.1 Hierarchical Model

The hierarchical model is the oldest of the three models. This model structures data so that each element is subordinate to another in a strict hierarchical manner.

The hierarchical model represents 1:n relationship between record types (see Fig. 3.6). One record type (the 1 in 1:n relationship) is designated as the "parent" record type. In this parent child relationship, a child record type can have only one parent record type but a parent record may have several child record types.

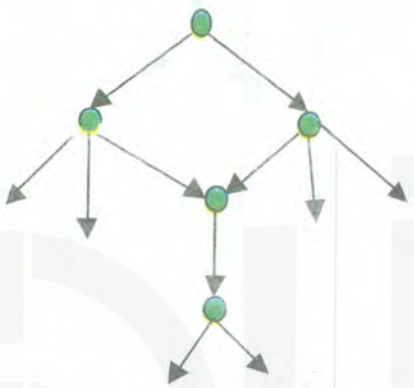


Fig. 3.6: A Hierarchy (1:n relationship)

The hierarchical model is implemented by storing data in physical adjacency and using pointers to the dependent child records. The model suffers from undesirable data redundancy.

An example of the hierarchical model is illustrated in Fig. 3.7. Here hierarchy has been shown between two record types—one having fields like name, address, profession, account number and the other giving account number and balance. Redundancy has been shown by an arrow.

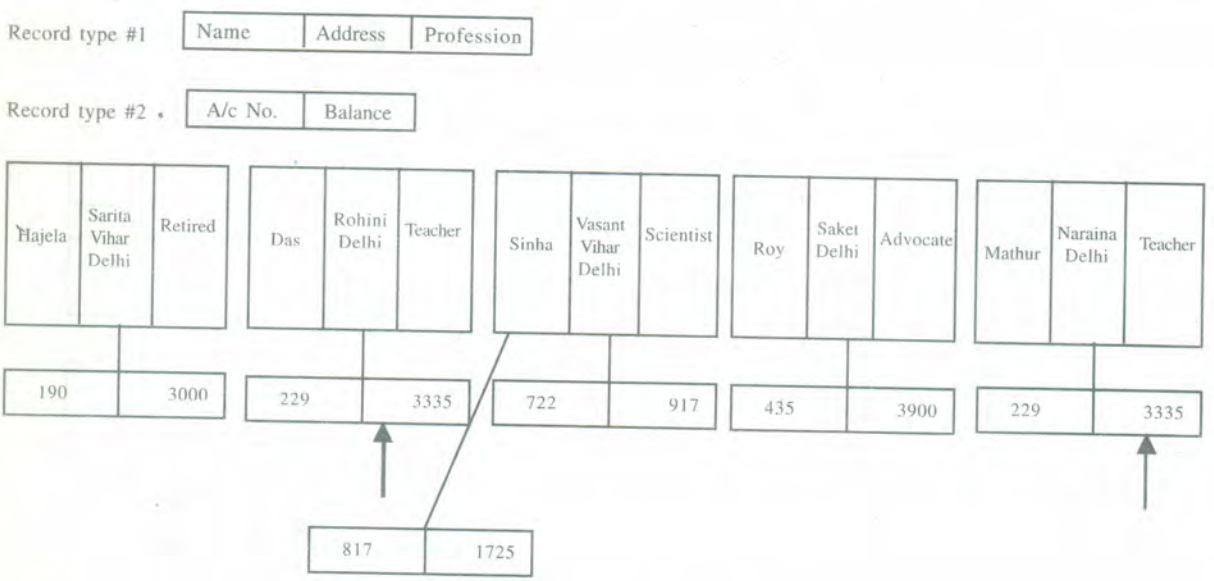


Fig. 3.7: An Illustration of the Hierarchical Model

Examples of commercial implementation of the hierarchical model are IBM's IMS database management system and CDS/ISIS – a popular database management system for bibliographic applications.

3.7.2 Network Model

In the network model the restriction that a child can have only one parent record is removed. The network supports n:m relationship (see Fig.3.8). A network can be hierarchy (1:n being a special case of n:m) but a hierarchy cannot be a network.

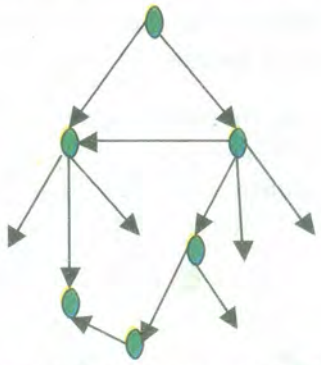


Fig. 3.8: A Network (n:m relationship)

The network model is implemented with various pointer schemes. Since a network is an extension of hierarchy, the semantic properties of the network model are similar to those of the hierarchical model.

The network model is illustrated in Fig.3.9. The example given makes use of the same record types as in the hierarchical model with the difference that the first record type has pointers to the A/c No.of the second record type. A pointer may itself point to a number of pointers (called arrays).

Cullinet's IDMS is an example of commercial database management system based on the network model.

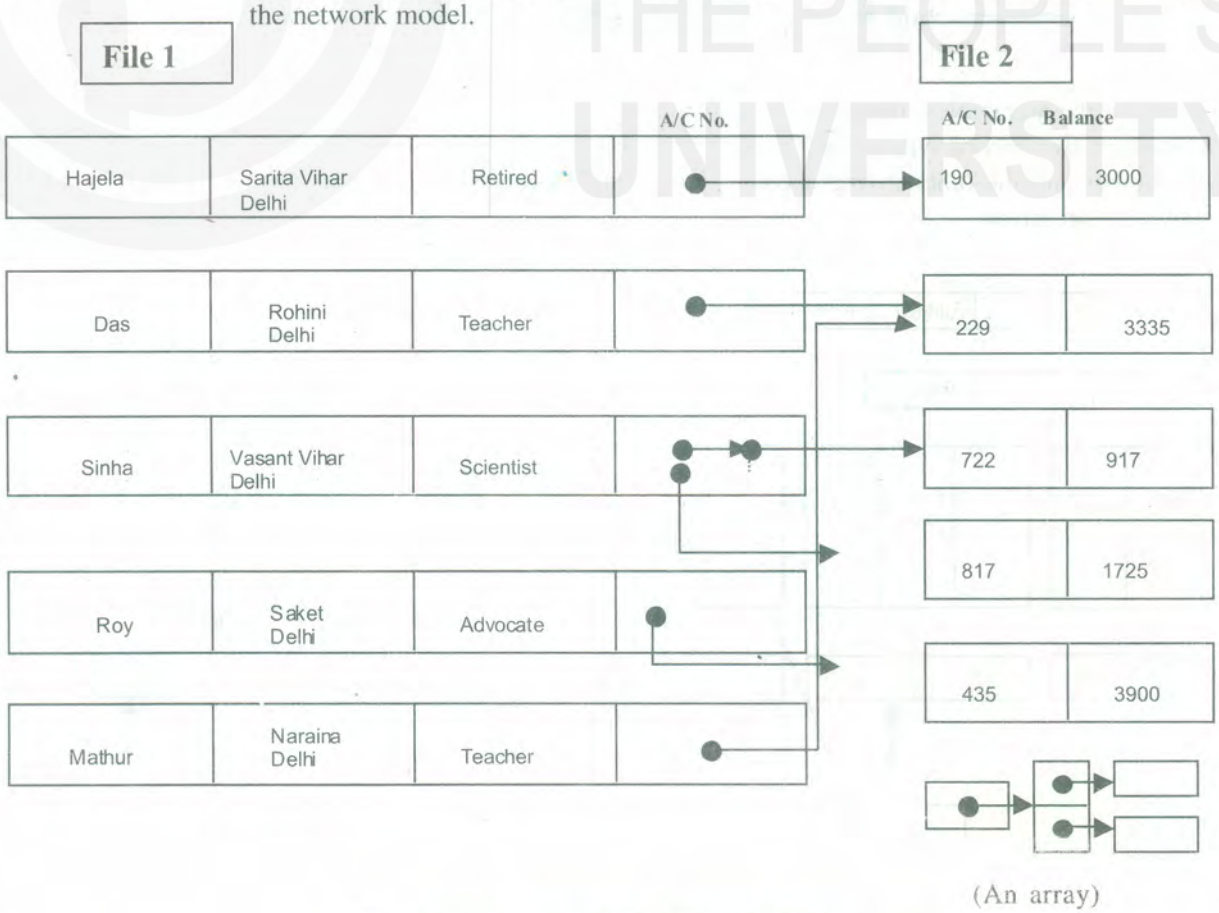


Fig. 3.9: An Illustration of the Network Model

3.7.3 Relational Model

The relational model maintains data in tabular form, which users find comfortable and familiar. The model is based on well-developed mathematical theory of relations from which it derives its name. The name has nothing to do with the fact that the data stored in the relations is related (which usually is).

The relational model supports 1:1, 1:n, n: 1 and n:m relationships. A significant aspect of the relational model is that the relationships between data items are not explicitly stated by pointers. Instead, it is up to the DBMS to deduce the relationships from the existence of matching data in different tables. The absence of physical links provides a more flexible data manipulation environment.

An illustration of the relational model is given in Fig. 3.9. Here the relationship between the two tables has been captured by repeating a column (A/c No.) in the first table.

Table name : Customer

Name	Address	Profession	A/C No.
Hajela	Sarita Vihar Delhi	Retired	190
Das	Rohini Delhi	Teacher	229
Sinha	Vasant Vihar Delhi	Scientist	817
Sinha	Vasant Vihar Delhi	Scientist	722
Roy	Saket Delhi	Advocate	435
Mathur	Naraina Delhi	Teacher	229

Table name : Account

A/C No.	Balance
190	3000
229	3335
817	1725
722	917
435	3900

Column repeated to establish relationship

Fig. 3.10: An Illustration of the Relational Model

ORACLE, INGRESS, and SYBASE are some of the well-known commercial database management systems based on the relational model.

3.7.4 Object-oriented Model

The object-oriented data model facilitates handling of objects rather than records. In an object-oriented model an entity is represented as an instance (object) of a class that has a set of properties and operations (methods) applied to the objects. A class represents an abstract data type and is a shell from which one can generate as many copies (called instances) as one wants. In object-oriented approach, the behaviour of an object is a part of its definition. The behaviour is described by a set of methods. The set of methods offered by an object to the others defines the object interface. A class and hence an object may inherit properties and methods from related classes. Objects and classes are dynamic and can be created at any time.

Viewing the data as objects instead of as records provides more flexibility and removes the need to normalise data.

Fig. 3.11 gives comparison between the conventional database approach and object-oriented database approach.

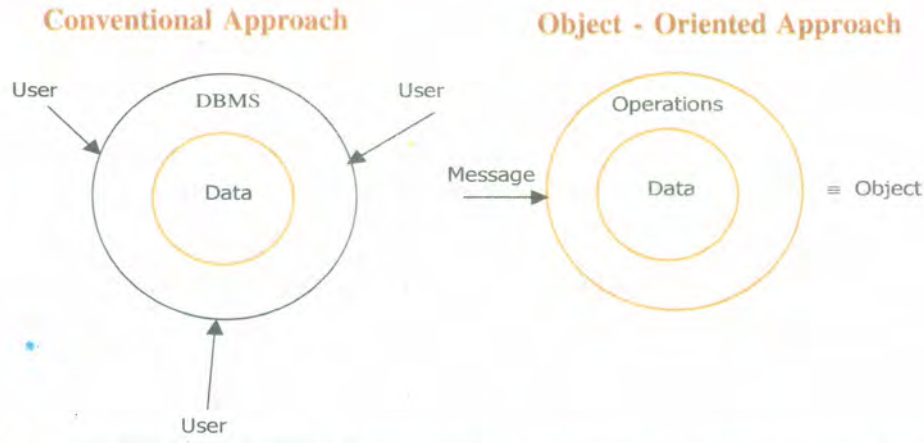


Fig. 3.11: Comparison between Conventional and Object-oriented Database Approaches

Some of the object building blocks are defined below:

- Objects: An object is an entity, real or abstract, that has *state, behaviour* and *identity*. The state of an object is represented by its *attributes* and their values. The behaviour of an object is represented by its operations or *methods*.
- Messages: Objects communicate with each other through messages. A message determines what operation is to be performed by an object. A message specifies an *operation name* and a list of arguments.
- Classes: A class is a set of objects that share common attributes and behavior. Each object is an instance of some class.

The object-oriented approach emphasises incremental software development. The underlying principle of this approach is:

- Grow software, don't build it;
- Build components rather than a whole system; and
- Assemble a basic system and then enhance it.

Smalltalk, C++, Java and Object Pascal/Delphi are the object-oriented programming languages used in this approach.

Self Check Exercise

4) Why have RDBMS found wider application than other data models? Give a few examples of RDBMS.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.8 RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)

Relational database management systems have been evolving since the relational data model was first proposed in 1970 by Edgar F. Cod of IBM. They have become de facto international standard in database management. Despite great advances in the object-oriented database management systems, relational systems are likely to remain in vogue for quite some time.

Let us define some of the terms used in relational technology.

A relational database is collection of data belonging to a system spread over a number of relations.

As pointed out earlier, relation is a mathematical concept. Mathematically, a relation is a subset of the Cartesian product of a list of domains. A domain represents a set of values an attribute can assume (e.g., numeric, character, etc.).

The degree of a relation determines the number of attributes in the relation. The number of tuples in a relation is called its cardinality. For example, the customer table in Fig. 3.9 has four attributes and six tuples and hence the degree of this relation is four and cardinality six.

A relational database management system (RDBMS) is a collection of programs that help to organise and manipulate data in a relational database.

The following terms are used interchangeably in RDBMS jargon:

- relation, table, database, file
- attribute, column, field
- tuple, row, record
- domain, range, type

3.8.1 Characteristics of a Relation

A relation exhibits the following characteristics:

i) A relation is always named.

ii) Each column of a relation has a unique name (although columns of different relations may have the same name).

iii) The order of the columns in a relation is of no consequence.

iv) There cannot be two identical tuples in a relation.

v) The order of the tuples in a relation is of no consequence.

vi) Each column of a relation has a domain specification.

vii) There may be more than one column in a relation having the same domain specifications.

viii) A column in a relation cannot have more than one domain specification.

ix) Each column contains values about the same attributes and each table cell value (intersection of a row and a column) must be single-valued.

The structure of a relation i.e., set of attributes without any values assigned to them is called the relation scheme. A tuple of a relation with values assigned to its attributes is called a relation instance.

A table is generally represented by its relation scheme which is denoted by table name followed by attribute names given in brackets. The relation schemes of tables shown in Fig. 3.10 are :

CUSTOMER (Name, Address, Profession, A/c No.)

ACCOUNT (A/c No., Balance).

The primary key attribute is underlined in the relation scheme.

3.8.2 Keys and their Functions

Keys play an important role in relational database management systems. They serve two basic purposes:

i) Identification of tuples

ii) Establishing relationship between tables

Keys (data items) can be made up of single attributes called single key attributes or multiplekey attributes called multikey attributes. Keys formed by multiple attributes are also called *composite or concatenated* keys.

A *superkey* is a set of attributes which taken collectively allows us to identify uniquely an entity in an entity set. If any key is a superkey, a superset of superkey is also a superkey. (If X1 is a subset of a set X, the X is called superset of X1).

The smallest superkey, which is also called minimal key, is a key such that no proper subset of it is a superkey. One of the minimal keys is chosen as the *primary key*. The keys in the set of minimal keys are called *candidate* or *alternate* keys. It is up to the database designer to select one of the candidate keys as a primary key.

A primary key is an attribute or a combination of attributes that uniquely identifies a record while a *secondary key* does not identify a record uniquely. A secondary key identifies all the records corresponding to the key value.

A *foreign key* is an attribute or a combination of attributes that is used to link tables. In relational database management systems foreign keys are used as linking pins between tables. Foreign keys represent links to the primary keys.

Primary and foreign keys are critical in relational database management systems due to their contribution in defining integrity rules. A paramount guideline in relational systems is that a primary key or any attribute participating in a composite primary key of a relation cannot have null value. This rule is called entity integrity.

There is another integrity rule, which pertains to foreign keys. According to this rule an attribute that is a foreign key in one table must be a primary key in another table. This rule is known as referential integrity.

3.8.3 Criteria for a DBMS to be Relational

Distinguishing a truly relational DBMS from relational-like systems assumes importance in view of the fact that many new DBMS packages are being labelled as "relational". The minimum conditions for a system to be called relational are:

i) The data should be represented in the form of tables.

ii) Any pointer mechanism should be transparent to the DBMS users.

iii) The system should support relational algebra operators of SELECT, PROJECT and JOIN.

Any system, which fulfills these three criteria, is called *minimally relational*. A system which satisfies only the first two conditions, is not a relational system and is called *tabular DBMS*.

For a DBMS to be fully relational it should additionally support both entity and referential integrity rules and implement all relational algebra operations.

The founder of relational database theory, E.F. Codd outlined 12 rules that define a fully relational database system. These rules are based on the premise that a relational database management system should be able to manage databases entirely through its relational capabilities.

3.9 NORMALISATION OF RELATIONS

Normalisation of relations is an important aspect in database design, which deals with semantics of the data. It is a technique that structures data in such a manner that there are no anomalies in the database. Anomalies refer to undesirable side effects which can occur in relations resulting in poor database design. The process of normalisation usually involves decomposing a relation into two or more relations with fewer attributes i.e., taking a vertical subset of the parent relation. The criteria used to split relations that determine the levels of normalisation are called *normal forms*.

It ought to be noted that normalisation is primarily aimed at preventing or reducing data maintenance problems rather than improving the retrieval efficiency. Converting relations to normal forms and utilising the join of the decomposed relations to retrieve data that could have been retrieved from one original table does not augur well for retrieval speed. Thus, while normalising relations we are sacrificing retrieval speed to improve the integrity, consistency and overall maintenance of data stored in the database.

As indicated earlier, normalisation of relations removes anomalies in the database. The anomalies can be categorised as:

- Insertion anomalies
- Deletion anomalies
- Update anomalies

An *insertion anomaly* occurs when we are unable to insert a tuple into a table. Such a situation can arise when the value of primary key is not known. As per the entity integrity rule, the primary key cannot have null value. Therefore, the value/s corresponding to primary key attribute/s of the tuple must be assigned before inserting the tuple. If these values are unknown, the tuple cannot be inserted into the table.

In case of a *deletion anomaly*, the deletion of a tuple causes problems in the database. This can happen when we delete a tuple, which contains an important piece of information, and the tuple being the last one in the table containing the information. With the deletion of the tuple the important piece of information also gets removed from the database.

An *update anomaly* occurs when we have a lot of redundancy in our data. Due to redundancy, data updating becomes cumbersome. If we have to update one attribute value, which is occurring a number of times, we have to search for every occurrence of that value and then change it. The anomalies will be further elaborated when we discuss the normal forms Before we proceed with discussion on normalisation it would be useful to understand the concept of dependencies.

3.9.1 Dependencies

A dependency refers to relationship amongst attributes. These attributes may belong to the same relation or different relations. Dependencies can be of various types viz., functional dependencies, transitive dependencies, multivalued dependencies, join dependencies, etc. We shall briefly examine some of these dependencies.

Functional Dependency (F.D) – Functional dependency represents semantic association between attributes. If a value of an attribute A determines the value of another attribute B, we say B is functionally dependent on A. This is denoted by:

A → B and read as "A determines B" and A is called the determinant.

It should be noted that when a data item (or collection of data items) determines another data item, the second data item does not necessarily determine the first. An attribute is said to be fully functionally dependent on a combination of attributes if it is dependent on the combination of attributes and not functionally dependent on any proper subset of the combination. To illustrate functional dependency let us consider a relation with the following relation scheme :

BOOK (Book-id, Subject, Year of publication, Price). Book-id (Book identifier or identification number) is the primary key of the relation and hence determines subject, year of publication and price.

We can represent it as:

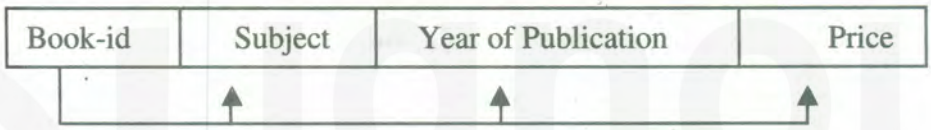
Book-id → Subject

Book-id → Year of publication

Book-id → Price

This can also be shown graphically in the form of a dependency diagram :

Book



Transitive Dependency – Transitive dependency is a form of intermediate dependency. For example, if we have attributes or groups of attributes A, B and C such that A determines B and B determines C i.e.,

A → B

B → C

Then we say a transitive dependency represented by A → B → C exists between A and C.

Let us consider a relation FACULTY.

FACULTY (Faculty-id, Name, Department, Office, Salary)

In this relation let us presume that each department has its own office building. Then beside others we have the following functional dependencies:

Faculty-id → Department (by virtue of Faculty-id being the primary key)

Department → Office (by virtue of the enterprise rule or constraint imposed by us on the relation)

Thus, we have the following transitive dependency in the relation :

Faculty-id → Department → Office

Multivalued Dependency — Multivalued dependency refers to m:n (many-to-many) relationships. We say multivalued dependency exists between two data items when one value of the first data item gives a collection of values of the second data item i.e., it multidetermines the second data items.

Join Dependency — If we decompose a relation into smaller relations and the join of the smaller relations does not give us tuples as in the parent relation, we say the relation has join dependency.

Let us consider a relation SAMPLE with the functional dependencies as shown:

A	B	C	F.D :	A → B
a1	b1	c1		
a2	b2	c3		C → B
a3	b1	c2		
a4	b2	c4		

Let us now decompose this relation into two relations SPLIT 1 (A, B) and SPLIT 2(B, C). The functional dependencies will remain the same in the relation SAMPLE.

SPLIT 1

A	B
a1	b1
	F.D.: A →
a2	b2
a3	b1
a4	b2

SPLIT 2

B	C
b1	c1
	F.D.: C → B
b2	c3
b1	c2
b2	c4

Now, if we join the relations SPLIT 1 and SPLIT 2 over common attribute B we get the relation SAMPLE 1.

SAMPLE 1

A	B	C
a1	b1	c1
a1	b1	c2*
a2	b2	c3
a2	b2	c4*
a3	b1	c2
a3	b1	c1*
a4	b2	c3*
a4	b2	c4

We can see from the relation SAMPLE 1 that it has four additional (spurious) tuples (shown by asterisk), which were not present in the original relation SAMPLE. This type of join is called lossy join because the information content of the original table is lost. This has occurred since the join attribute was not the determinant in the original relation and hence it should not have been decomposed the way it was done. We say a join dependency exists in this case. If we decompose a relation and join the constituent relations over the determinant attribute, we get lossless join.

For example, consider the relation SAMPLENEW and split it into SAMPLENEW1 and SAMPLENEW2 as given below:

SAMPLENEW

X	Y	Z	
x1	y1	z1	
x2	y2	z2	F.D: X→ Y
			X→ Z
x3	y2	z1	
x4	y1	z2	

SAMPLENEW1

X	Y	F.D: X → Y
x1	y1	
x2	y2	
x3	y2	
x4	y1	

SAMPLENEW2

X	Z	F.D: X → Z
x1	z1	
x2	z2	
x3	z1	
x4	z2	

The join of SAMPLENEW1 and SAMPLENEW2 gives the original table SAMPLENEW without any spurious rows because the attribute X over which the table is decomposed is the determinant attribute.

3.9.2 Normal Forms

The technique of normalization is based on the analysis of functional dependencies between attributes. Taking into account the total scenario, enterprise rules and semantic constraints, all the functional dependencies in the relations are captured and the relations transformed into proper normal forms. A normal form represents the level of normalization of a relation. Normalization proceeds by converting a relation from a lower normal form to higher normal form.

The norm forms are:

> First normal form (1NF), second normal form (2NF), third normal form (3NF), Boyce-Codd normal form (BCNF), fourth normal form (4NF), fifth normal form (5NF) and the highest normal form which is called domain/key normal form (DK/NF).

E.F. Codd had outlined the first three normal forms, which we shall discuss in details. Most of the commercially available DBMS are normalized up to 3NF or BCNF.

Normal forms build on each other i.e., if a relation is in 3NF, it is also in 1NF and 2NF.

The first normal form (1NF). A relation is in the first normal form if it can be represented as a flat file i.e., the relation contains single values at the intersection of each row and column.

In other words, each attribute value in a tuple is atomic i.e., non-decomposable.

Let us consider a relation PERSON that is not in 1NF.

PERSON (Name, Age, Gender, $C_{Name, Age}$)

In this relation $C_{Name, Age}$ pertains to the attribute dependent child with multiple values (name and age). To convert this relation into 1NF, it should be decomposed into two relations as follows:

PERSON (Name, Age, Gender)

DEPENDENT (Name, C_{Name}, C_{Age})

The second normal form (2NF). A relation is in the second normal form if it is in 1NF and every non-key attribute is fully functionally dependent on the primary key. The second normal form pertains only to relations with composite primary key. In case a relation is in 1NF and has a single-attribute primary key, it is automatically in the 2NF.

To explain the second normal form let us take the example of the following relation.

COURSE (Course-id, Course-name, Class-number, Student-id, Faculty-id).

In this relation Course-id, Student-id and Faculty-id form a composite primary key.

The following functional dependencies can be noticed in this relation.

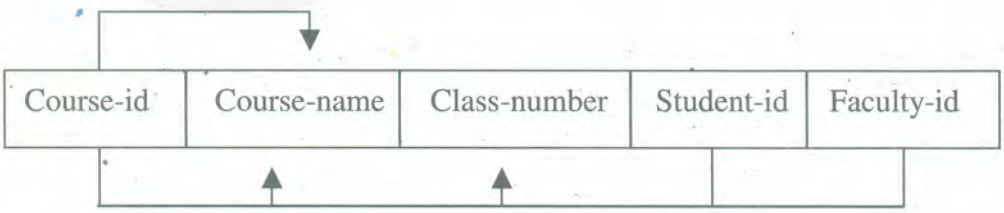
Course-id, Student-id, Faculty-id → Course-name

Course-id, Student-id, Faculty-id → Class-number

On examining the relation semantically we observe that Course-id uniquely determines Course-name i.e.,

Course-id → Course-name

This means that Course-name is not fully functionally dependent on the primary key. Therefore the relation is not in the 2NF. The dependency diagram of the relation can be represented as follows :



To convert this relation into the 2NF, we decompose it into two relations as follows:

COURSE (Course-id, Class-number, Student-id, Faculty-id)

COURSE-TITLE (Course-id, Course-name)

The decomposition is done by extracting the attribute that caused the problem from the relation and creating a new relation with this attribute.

Let us examine these relations (normalised and unnormalised) on the basis of anomalies described earlier.

COURSE (unnormalised)

Course-id	Course-name	Class-number	Student-id	Faculty-id
C701	Computer science	7	S009	B03
C701	Computer science	8	S008	G04
C702	Library automation	7	S009	A01
C702	Library automation	8	S006	G04
E500	Microeconomics	7	S009	P02
E501	Macroeconomics	7	S001	P02
M200	Management studies	7	S001	V01

The normalised relations obtained from the above relation are :

COURSE

Course-id	Class-number	Student-id	Faculty-id
C701	7	S009	B03
C701	8	S008	G04
C702	7	S009	A01
E500	8	S006	G04
E501	7	S007	P02
M200	7	S001	V01

COURSE-TITLE

Course-id	Course-name
C701	Computer science
C702	Library automation
E500	Microeconomics
E501	Macroeconomics
M200	Management studies

Insertion: In the case of unnormalised relation, suppose a new course named 'Computer networks' has to be introduced. We cannot enter the Course-name in the relation, since we have no means to ascertain Student-id and Faculty-id at the time of introduction of the course and these attributes, which along with Course-id form the primary key, cannot be assigned null values. However, in the normalised relation COURSE-TITLE this problem does not arise.

Deletion: Suppose in the unnormalised relation the student with Student-id S001 leaves the course. This student being the only student in the particular course, deletion of the

tuple would result in loss of information and we will have no way to know that the course 'Management Studies' exists. In normalised relation the deletion of this tuple does not result in information loss because the information about the course name exists in the second relation.

Updating: Let us assume that the name of the course 'Library automation' changes to 'Library management'. In the unnormalised relation we will have to change the information in two tuples while in the normalised relation only one tuple would require to be updated. This may not appear to be a significant gain but if the database size is large the problem can have serious repercussions.

Third normal form (3NF). A relation is in the third normal form if it is in second normal form and contains no transitive dependencies. This normal form is considered to be the most important normal form.

To illustrate 3NF, let us consider the following relation:

FACULTY (Faculty-id, Faculty-name, Department, Gender, Salary, Office)

Let us also assume that each department has its office in one building.

This relation has the following functional dependencies by virtue of Faculty-id being the primary key.

Faculty-id → Faculty-name

Faculty-id → Department

Faculty-id → Gender

Faculty-id → Office

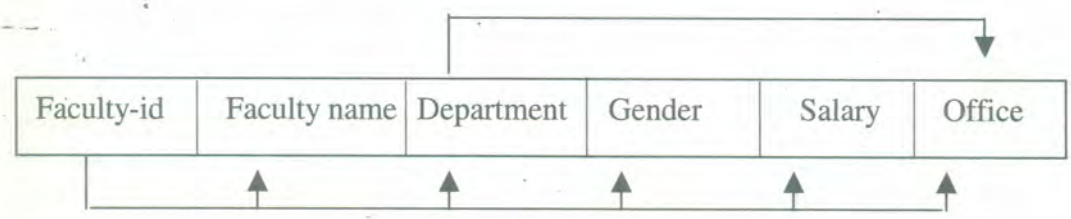
We also know from the semantics of this relation that

Department → Office

Therefore, we have a transitive dependency as shown below :

Faculty-id → Department → Office

Thus dependency diagram of the relation can be represented as :



To convert this relation into the 3NF we shall have to decompose it into two smaller relations. The new relation OFFICE-NAME has the attribute office which caused transitive dependency and its determinant. The decomposed relations are :

FACULTY (*Faculty-id,* Faculty-name, Department, Gender, Salary)

OFFICE-NAME (*Department,* Office)

With some tuple values in these relations let us examine them for anomalies, which are used to test relations.

FACULTY (unnormalised)

Faculty-id	Faculty-name	Department	Gender	Salary	Office
B03	Bindra	Computer Science	M	4000	Birla Block
G04	Ganguly	Computer Science	M	4500	Birla Block
A01	Arora	Computer Science	F	3450	Birla Block
P02	Pandey	Economics	F	3500	Kanishka Block
V01	Vohra	Mathematics	M	4500	Ashoka Block
B01	Bansal	Physics	M	3000	Ashoka Block

The normalized relations are:

FACULTY

Faculty-id	Faculty-name	Department	Gender	Salary
B03	Bindra	Computer Science	M	4000
G04	Ganguly	Computer Science	M	4500
A01	Arora	Computer Science	F	3450
P02	Pandey	Economics	F	3500
V01	Vohra	Mathematics	M	4500
B01	Bansal	Physics	M	3000

OFFICE-NAME

Department	Office
Computer Science	Birla Block
Economics	Kanishka Block
Mathematics	Ashoka Block
Physics	Ashoka Block

Insertion: Let us assume that a new department whose faculty has not yet been finalized is created. We cannot enter this information in the unnormalised relation because we do not know value of the primary key (Faculty-id). To assign values to Faculty-id corresponding to new department we must know the values of other attributes (Faculty-name, Gender, Salary, Office). This problem does not exist in the normalised relation.

Deletion: Suppose a faculty member Vohra of mathematics department leaves the faculty. If we delete tuple corresponding to this Faculty-name in the unnormalised relation, the information that mathematics department exists is also lost. However, this does not happen in the normalised relations.

Update: If the office of the Computer Science department shifts from Birla Block to Nehru Block, 3 tuples need to be updated in the unnormalised relation while only one tuple would be modified in normalised relation.

Boyce-Codd normal form (BCNF). Originally, normal forms stopped at the 3NF. However, research into dependencies led to higher normal forms. BCNF is an extension of the third normal form. It states that if a relation is in 3NF and all determinants are candidate keys, then it is in Boyce-Codd normal form.

The fourth normal form (4NF) refers to multivalued dependencies. A relation is said to be in the fourth normal form if it has only one multivalued dependency.

The fifth normal form (5NF) is also called project join normal form (PJNF). If a relation is in 5NF we should be able to join the projections (decompositions) of the relation and reconstruct the original relation without any information loss.

The domain key normal form (DK/NF) is considered the highest normal form. A relation is in DK/NF if every constraint can be inferred by simply knowing the set of attribute names and their underlying domain along with their set of keys. Thus in DK/NF only two types of constraints are allowed – domain constraints and key constraints. If these constraints are fully enforced, other constraints (dependencies) are removed and no anomalies can occur in the relation.

The process of normalisation from lower to higher normal forms has been illustrated in Fig. 3.12.

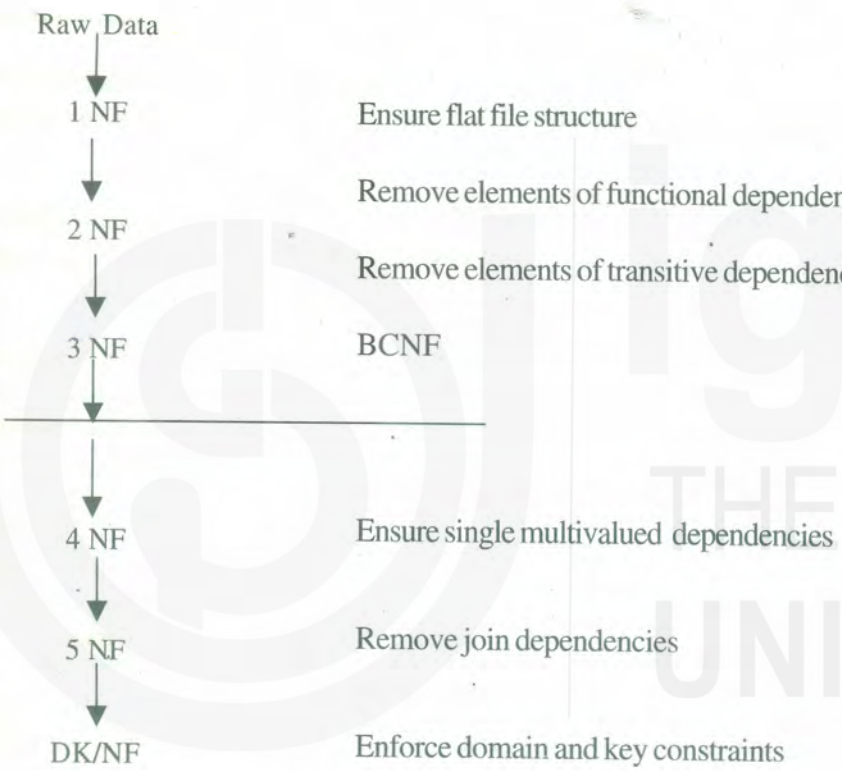


Fig. 3.12: Normalisation Process

Self Check Exercise

5) What are the advantages and disadvantages of normalisation of relations ?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.10 DESIGNING DATABASES

Designing a database is a highly complex operation. Though it is relatively easy to identify a poorly designed database, there does not exist a unified approach which leads to the best design.

The database design should be flexible enough to meet the requirements of the maximum number of users to the fullest. Besides, the design should also anticipate, to a certain extent, future requirements and make provisions for them. This calls for some intuitiveness on the part of the database designer.

The process of designing a database is an interactive one, which means that initial database structure, changes with usage. However, with time the design tends to get stabilized. Usually, a person designated as database administrator (DBA) controls the design and administration of a database.

A broad step-by-step procedure for designing a database has been summarized below:

1) First of all, data to be represented in the database is determined. For this, information needs of the users are studied in detail. Based on the information requirements anlysis, entities of interest are identified and their attributes examined.

2) An E-R model of the database representing conceptual schema is drawn. This is the most important stage in the database design process. E-R diagram, which depicts entitites and their relationships, should be as comprehensive as possible.

3) The E-R model is mapped into a selected database structure (hierarchical, network, relational or any other model). In case a relational model is chosen, tables corresponding to entities and their relations are finalised. The process of normalisation is invoked to check the tables and reshape them if necessary.

4) An empty database is created using DBMS commands (create TABLE, INDEX, etc.). A data dictionary, which defines data item names and their internal storage format, is also created by DBMS. The database design process up to step 3 is system-independent. The process becomes system-dependent after that.

5) The database is populated. This involves inserting data into the empty database. If data to be inserted is available in machine-readable form, data loading utility of DBMS can be utilised.

6) The performance of the database is closely monitored to ascertain whether any tuning is required. Flexibility and speed of access are critically evaluated. The database is also examined for data maintenance problems.

7) The feedback of the users on the database functionality is analysed and changes in the structure made to optimise the usage.

3.11 DISTRIBUTED DATABASE SYSTEMS

3.11.1 Architecture of Distributed Databases

A distributed database is a single logical database, which is fragmented, and the fragments spread across computers at different locations that are interlinked by a data communication network to provide an integrated access to the data. A distributed database environment requires the data to be shared. A distributed database gives geographical data independence i.e., a user requesting data need not know at which site the data is located. This property is often referred to as location transparency and each local site is called a node.

Architecture and schematic representation of a distributed database is shown in Fig.3.13

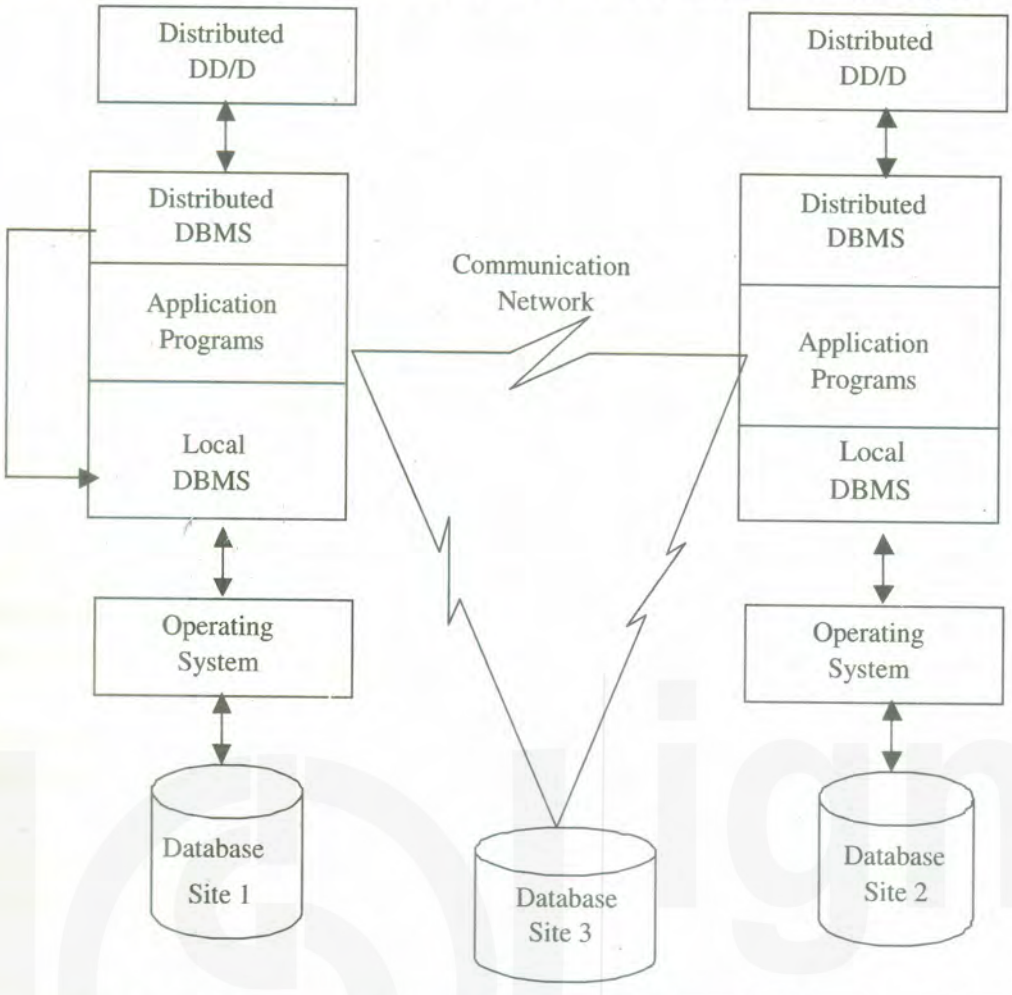


Fig. 3.13: Architecture and Schematic Representation of a Distributed Database

As is clear from Fig.3.13, each site has a local DBMS as well as a copy of the distributed DBMS. Distributed data dictionary/directory (DD/D) stores information on location of data in the network as well as data definitions. Request for data is first checked from the distributed data dictionary/directory for location of the required data. In case the data is available at the local site, the distributed DBMS forwards the request to local DBMS for processing. If the request involves data from other sites, the distributed DBMS routes the request to these sites.

When different nodes in a distributed database have mixed DBMSs (i.e., node 1 may have relational DBMS and node 2 network DBMS), then the distributed DBMS capable of handling such environment is called heterogeneous distributed database management system.

Distributed database management exploits all advantages of centralised and decentralised processing. A decentralized database, like a distributed database, is also stored on different computers at multiple locations but in this case the computers are not interconnected and hence the data cannot be shared.

3.11.2 Justifications and Options for Distributing Data

The justifications for distributing data can be summed up as :

- A distributed database provides increased reliability and availability. Compared to a centralised system which on failure becomes unavailable to all users, a distributed system will continue to function, though at a reduced level, even when a node fails.
- By encouraging local control of data at different sites, data integrity improves and data administration becomes easier.

- Distribution of data can improve access time if local data is stored locally. By locating data closer to the point of its use, communication cost can be reduced and query response time improved.
- Distributed system facilitates modular growth. New nodes hosting additional database fragments can be added to the system.

There are a number of options available for distributing data in a distributed database. These options include: i) data replication, ii) horizontal partitioning, iii) vertical partitioning and iv) combination of the above.

In case of **data replication** a copy of the database is stored at a few or all sites (full replication). Reliability, saving in telecommunication charges and faster response are the advantages of this option. But additional storage requirements and difficulty in propagating updates form the basic drawbacks. This option is suitable in case updates are infrequent and database interaction is restricted to read-only. CD-ROM (compact disk read only memory) offers excellent medium for replicated databases.

Horizontal partitioning of a database involves distributing rows of a relation to multiple sites. New relations (partitions) with the requisite rows are created for this purpose. The original relation can be reconstructed by taking the union of the new relations. Horizontal partitioning can optimize performance by storing fragments of the database at the sites where they are most used.

On **vertical partitioning** of a database, selected columns of a relation are projected into new relations, which are stored at different sites. The main criterion for vertical partitioning is specific data item requirements at individual sites.

A combination of the mentioned options of data distribution may be used depending upon the needs of the distributed system. The basic principle, which one must keep in mind, is that data should be stored at sites where it will be most frequently used.

3.12 DATABASE SYSTEMS FOR MANAGEMENT SUPPORT

Database systems for management support are broadly referred to as Management Information Systems (MIS). However, for different levels of management database systems have been categorised based on management functions and expected outputs. Fig.3.14 illustrates a hierarchy of information systems corresponding to the three levels of management.

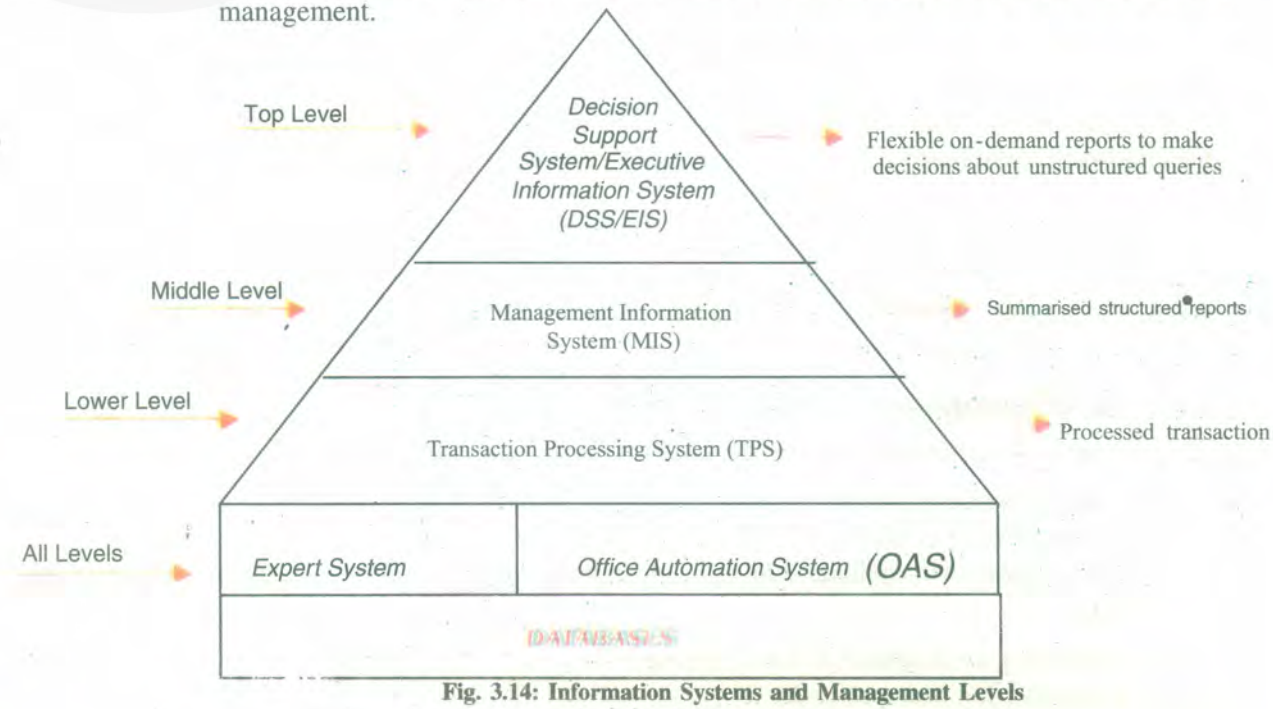


Fig. 3.14: Information Systems and Management Levels

As is clear from the figure, for lower level of managers, transaction-processing systems (TPS) yielding processed transactions (bills, orders etc) suffice. Middle level managers need management information systems providing summarised structured reports. At the top-level decision support systems (DSS) or executive information systems (EIS) capable of providing brief on-demand reports about unstructured queries are required. Office automation systems and expert systems are used by all levels including non-management.

Self Check Exercise

6) What are the advantages of distributed databases?

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.13 ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Artificial intelligence (AI) is a group of related technologies that attempt to develop machines to emulate human-like qualities such as learning, reasoning, communicating, seeing and hearing.

In the early days of its development a computer was called "electronic brain" and since then efforts have been made to empower it with capabilities comparable with human brain.

However, there is a basic difference in the functioning of a computer and human brain. Whereas a computer processes numbers, human brain works on symbols. A conventional computer program uses algorithms i.e., well-defined step-by-step procedure to solve a problem while human thought process is not algorithmic and is based on symbolic processing. In symbolic processing information are represented using symbols rather than numbers. Human intelligence or thought process relies on heuristic methods for information processing. Heuristics of human thought process can be represented as shown in Fig. 3.15.

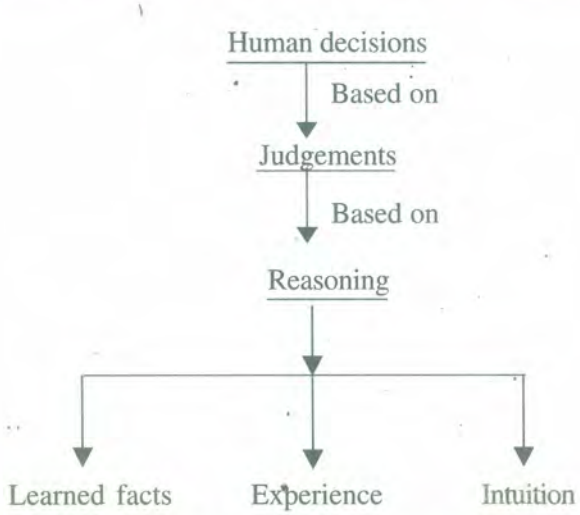


Fig. 3.15: Heuristics of Human Thought Process

The main areas of AI are:

- Robotics
- Perception systems
- Fuzzy logic
- Neural networks
- Genetic algorithm
- Natural language processing
- Expert systems

Robotics is a field that attempts to develop machines that can perform work normally done by people. The machines themselves are called robots. Perception systems are sensing devices that emulate the human capabilities of sight, hearing, touch and smell. Clearly, perception systems are related to robotics, since robots need to have some sensing capabilities. Fuzzy logic is a method of dealing with imprecise data and uncertainty, with problems that have many answers rather than one. Unlike classical logic, fuzzy logic is more like human reasoning. It deals with probability and credibility. That is, instead of being simply true or false, a proposition is mostly true or mostly false or more true or more false.

Fuzzy logic principles are applied in neural networks. Neural networks use physical electronic devices or software to mimic the neurological structure of human brain. Genetic algorithms refer to programs that use Darwinian principles of random mutation to improve themselves.

In natural language processing a user can interact with a computing system using everyday language rather than a structural command language. Natural language systems have an interface called natural language interface, which translates the natural language into the application languages (such as SQL). For a language to be understood there are two aspects namely, syntax and semantics which play a vital role. Syntax represents the rules for combining words to form phrases, clauses and sentences while semantics refers to word meanings and the way the concepts are expressed. A natural language interface is capable of analyzing syntax and semantics of a language. However, there are a number of limitations with the natural language systems at present, which have hindered their widespread use. With time the systems are likely to get more refined and popular.

An expert system can be defined as AI program designed to represent human expertise in a specific domain. Typical expert system application areas include diagnosis, planning, instruction and managment, monitoring and design.

At the heart of an expert system is a *knowledge base, which* contains facts (basic data), rules and other information pertaining to a particular domain. Facts represent accepted knowledge in a certain field (normally possessed by experts) and rules – a collection of IF – THEN statements also called "rules of thumb" (heuristics) for drawing inferences: IF such-and-such is true, THEN assume that so-and-so is true.

Unlike databases, which store explicit information, knowledge bases with their IF-THEN rules can infer additional information not directly stored in the basic data. L1SP (list processing) and PROLOG (programming in logic) are the languages most commonly utilised by knowledge bases.

The overall process of developing an expert system is called K*nowledge Engineering*. In order to map human knowledge to computer knowledge, one must understand the nature of knowledge. Knowledge can be grouped into: procedural knowledge, declarative knowledge, semantic knowledge and episodic knowledge. Procedural knowledge includes skills an individual knows how to perform. It refers to knowing how to do something. Declarative knowledge represents information that can be verbalised or expressed. It states facts about the world. Semantic knowledge is a deep-level knowledge that reflects cognitive structure, organisation and representation. Episodic knowledge represents information that has been chunked or compiled episodically. It refers to experiential information of an expert.

The expert systems are made up of three basic components: the knowledge base, an inference engine and a user interface. A specialist called knowledge engineer interviews the experts for the domain and encodes their knowledge (in the form of rules) into the knowledge base. The inference engine includes programs that are used to control how the rules in the knowledge base are used or processed. The user interface facilitates communication or interaction between the expert system and end user. Components of an expert system are represented schematically in Fig.3.16.

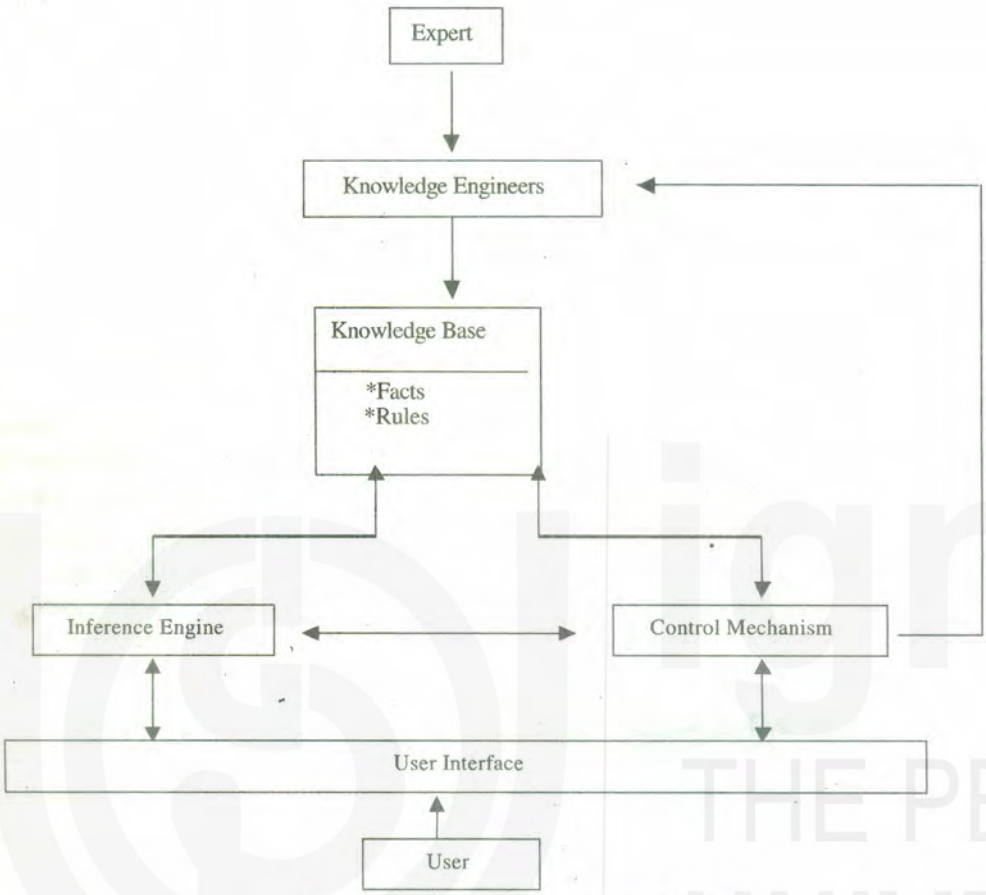


Fig. 3.16: Components of an Expert System

On request for a piece of specific information from a user, the system looks for data and rules in the knowledge base. The inference engine decides which rules to fire and how they should be applied and also when to complete the querying process and provide answer. User interface may prompt more input from a user to help the system respond effectively to the query.

The use of expert systems is growing rapidly. However, human experts are likely to remain in control using the expert systems as job aid.

Self Check Exercise

7) What is the difference between a knowledge base and a database?

8) What is artificial intelligence (AI)? List some of the areas of its application.

Note: i) Write your answers in the space given below.

ii) Check your answers with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

3.14 SUMMARY

This Unit discusses some of the basic issues in database management systems and provides essential background to understand the functionality of these systems.

The database architecture (three-layer) is a convenient tool for the user to visualize the schema levels in a database system. This architecture makes it easier to achieve data independence – both physical and logical.

The E-R model plays an important role in the conceptual database design. Evolutionary path of data models from hierarchical model to relational and then to object-oriented model has brought about tremendous changes in database design techniques. Relational database management systems are by far the most popular. Normalisation of relations is an important aspect of database design aimed to remove anomalies in a database. It improves integrity and consistency of data, though it slows retrieval speed. Designing databases is a highly complex process. Usually a database stabilizes in design over a period of time with feedback from users.

Database technology is making tremendous advances and a variety of database systems have appeared on the scene in recent years. Distributed database systems, which provide geographical data independence, permit large databases to be distributed over a number of locations transparent to the users. Knowledge engineering with expert systems and knowledge bases is providing new tools catalysing socio-economic developments. The needs of the emerging information society critically depend on the advancement in database technology.

3.15 ANSWERS TO SELF CHECK EXERCISES

1) In database systems, data independence is an important concept. It refers to separation of data from the programs that use it. Data independence enables the data definition to be changed without altering the programs.

   Data independence can be physical, logical or geographical (distributive). Physical independence means that one can change the way the data is actually stored or accessed in the system without requiring changes in the application programs. Logical independence means that the database can be reorganised i.e., changes can be made in the conceptual level but still the same application programs can be used. Geographical independence implies that the application programs are not affected by the location of data i.e., whether the data is located on a local disk or on a remote file server. Data independence imparts great flexibility in handling data in database systems.

2) A schema is an overall conceptual or logical view of data in a database. Collection of all the tables in a database. The schema contains a list of all the fields, the field types, and the maximum and minimum acceptable values for each field, along with information about the structure of every row in every table of the database.

   A subset or transformation of the logical view of the database schema that is required by a particular user application program is called subschema. A subschema is an individual view of the database. Each individual user may have a separate view of the database depending upon the user requirements. Access to the entire database i.e., conceptual schema is generally not given to all the users. A view helps to provide access to only that part of the database, which a user actually needs. The purpose of the views is not only to rationalise database access but also to implement security aspects.

3) E-R diagram is a tool that models the relationships among the entities in a database. E-R diagram maps the conceptual schema and serves as a blue print for designing databases.

4) Relational database management systems (RDBMS) have become de facto international standard in database management. Despite great advances in the object-oriented systems and other systems; relational systems retain their wide acceptance.

   RDBMS have advantages over other data models in the fact that the relational model is based on well-developed mathematical theory of relations from which it derives its name. Application of mathematics imparts great strength to the relational model. The data in relational systems is represented in the form of tables which users find easier to handle. Examples of RDBMS are: ORACLE, SYBASE and INGRESS.

5) The advantages of normalisation of relations are that it enforces data integrity and removes anomalies in a database. It also minimises data redundancy and in general promotes accuracy and consistency of data in the database.

   Since the process of normalization involves decomposing a table into two or more tables, which are joined while retrieving data, the retrieval speed gets adversely affected.

6) The advantages of distributed databases are as given below:

   - Increased reliability and availability
   - Improved data integrity and data administration
   - Improved access time
   - Modular growth

7) A database stores explicit information while a knowledge base with IF-THEN rules can infer additional information not directly stored in the basic data. In a knowledge base inference rules allow relationships to be derived from the data.

8) Artificial intelligence refers to computer programs, which emulate human intelligence i.e., programs facilitating computers to perform tasks that require intelligence when performed by humans. Examples of such tasks are visual perception, understanding natural language, game-playing, theorem-proving, medical diagnosis and engineering design.

   AI methods have been successfully applied in areas like: knowledge-based systems (expert systems, knowledge bases, etc.), robotics, computer vision, machine translation, neural networks and others.

3.16 KEYWORDS

Access Method : The method used to store, find and retrieve the data from a database.

Artificial Intelligence : A branch of computer science that is attempting to develop systems to emulate human-like qualities such as learning, reasoning, communicating, seeing and hearing.

Data Independence : Separates the data from the program, which often enables data definition to be changed without altering the program.

Data Integrity : Keeping accurate data which means few errors and the data reflect the true state of a database.

Dependency : A dependency refers to relationship amongst attributes belonging to the relation or different relations.

E-R Diagram : Entity-Relationship Diagram. A diagram that shows associations (relationships) between entities.

Expert System : A system with a knowledge base consisting of data and rules that enables users to make decisions as effectively as an expert.

Foreign Key : A column in one table that is primary key in a second table. It does not need to be a key in the first table.

Knowledge Base : A knowledge base is an expert system's database of knowledge about a particular subject. This includes relevant facts, rules and procedures for solving problems. The basic unit of knowledge is expressed as an IF-THEN- ELSE rule.

Normalisation : The process of creating a well-behaved set of tables to efficiently store data, minimize redundancy and ensure data integrity.

Primary Key : A column or a set of columns that identify a particular row in a table.

Relation : A relation is a table.

Relationship : An association between two or more entities.

Schema : An overall conceptual or logical view of the relationships between the data in a database.

Subschema : A subset or transformation of the logical view of the database schema that is required by a particular user application program.

Transparent : In computing, it pertains to a process or procedure involving a user without the later being aware of its existence.

3.17 REFERENCES AND FURTHER READING

CISMOD 93. *Proceedings of the International Conference on Information Systems and Management of Data* (1993). Indian National Scientific Documentation Centre (INSDOC): New Delhi.

Courtney, James F. and Paradice, David B. (1988). *Database Systems for Management.* Toronto: Times Mirror/Mosby College Publishing.

Codd, E.F. (1990). *The Relational Model for Database Management*. New York: Wesley Publishing Company. Inc.

Curtin, Dennis P. [et al.] (1999). *Information Technology: The Breaking Wave*. New Delhi: Tata McGraw-Hill Publishing Company Limited.

Date, C.J. (1989). *Introduction to Database Systems*. New Delhi: Narosa Publishing House.

Delobel, Claude and Adiba, Michel (1985). *Relational Database Systems*. Amsterdam: Elsevier Science Publishers B.V.

Elmasri, Ramaz and Navathe, Shaukan B. (2000). *Fundamentals of Database Systems.* Asia:Pearson Education Asia.

MeFadden, Fred. R. and Hoffer, Jeffrey A. (1988). *Database Management.* California : The Benjamin/Cummings Publishing Company. Inc.

Martin, James (1988). *Principles of Database Management*. New Delhi: Prentice-Hall of India Private Limited.

McNurlin, Barbara C. and Spragu, Ralph H. (1989). *Information Systems Management in Practice.* New Jersey: Prentice-Hall.Inc.

McGraw, Karn, L. and Karan, Harbison-Briggs (1989). *Knowledge Acquisition: Principles and Guidelines*. New Jersey: Prentice-Hall. Inc.

Murray, Linda. A. and John Richardon, T.E. (1989*). Intelligent Systems in a Human Context.* Oxford : Oxford University Press.

O'Brien, James A (1997). *Introduction to Information Systems.* Singapore: Post, Gerld V. (2000). Database Management System. Delhi: Tata McGraw-Hill Publishing Company Ltd. *Irwin/McGraw Hill.*

Post , Gerald V.(2000). *Database Management Systems.* New Delhi.: Tata McGraw-Hill Publishing Company Limited.

Shepherd, Robert D. (2001). *Introduction to Computers and Technology*. New Delhi: Crest Publishing House.

Su Stanley, Y.W (1988). *Database Computers: Principles, Architectures and Techniques.* Singapore: McGraw-Hill Book Company.

Ullman, Jeffrey, D. (1991). *Principles of Database Systems*. New Delhi: Galgotia.

Williams Brain K. [et al.] (1999). *Using Information Technology.* Singapore: Irwin/ McGraw-Hill.

UNIT 4 DATABASE SEARCHING

Structure

4.0 OBJECTIVES

After completing this Unit you will be able to understand the:

- search process and the steps in the formulation of an effective search; and
- characteristic features of search languages including the various search techniques and devices that are available to enhance a search.

Database searching is used here to mean searches carried out with the objective of retrieving relevant information including Internet / Web searches, online database searches (e.g. online databases provided by database vendors such as DIALOG, BRS, etc), searching of CD-ROM databases and searching of OPACs. Throughout this text the terms 'Search Engines' and 'Search Languages' are used inter-changeably.

4.1 INTRODUCTION

For the purpose of this Unit we shall define ***data*** as raw data - numbers or letters representing facts about entities and stored in a computer. ***Data*** in a computer has little meaning in its stored state in the physical form or logical form since any fact without the context is not usable. ***Database*** is a store of ***data*** – a collection of data, which exists for the purpose of providing information. Although databases can be manual or computerised, for the purpose of this Unit ***database*** *is a collection of data, held in computer files,*

which is generally accessible to multiple users. At this stage it is important to distinguish a ***database*** from a ***database system***, all the components of which interact to gather, manipulate, manage, search, retrieve and deliver information. The commonly used Database Management Systems (DBMS) belong to one variety of database systems that are built to process highly structured data. Information retrieval is often, a loosely defined term and we shall use it here to refer largely to automated information retrieval systems. Even the word information can be very misleading. In the context of Information Retrieval (IR), information, in the technical sense used by Shannon, is not readily measured. It is perhaps, in many situations, more appropriate to describe the kind of retrieval by simply substituting 'document' for 'information'. However, 'information retrieval' has been widely accepted as a technical term and is generally used in literature to refer to the sort of activities described in the works of Cleverdon, Salton, Sparck Jones, Lancaster and others. An Information Retrieval System (also referred to as Text Retrieval System) represents a system built to process and retrieve information from textual records. Most information retrieval systems merely inform the user on the existence (or non-existence) and whereabouts of documents relating to his request. As we all know, the internal organisation of most textual entities varies from structured to less structured and even unstructured. Generally, the smallest meaningful unit of data in a database is a *record* and this corresponds to an entity / '*thing*' in the real world. In information retrieval systems the records contain surrogates of textual entities such as books, papers in periodicals, etc. The architecture of different types of database systems has developed in response to both varying data and data processing requirements. In this Unit the focus will be on Information Retrieval Systems (IRS), i.e., on accessing and searching textual databases with particular emphasis on bibliographic databases.

4.2 INFORMATION RETRIEVAL

The term '*Information Retrieval*' was coined over five decades ago by Calvin Mooers to refer collectively to all the processes involved in searching and retrieving information about documents in a database and relevant to a query. The basic process of an IR system are diagrammatically represented in Figure 4.

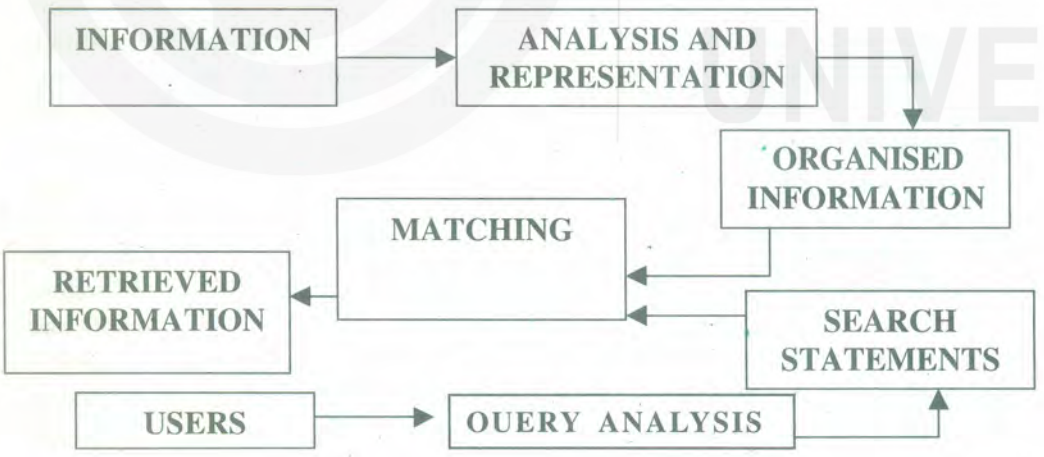


Fig. 4.1: Process of Information Retrieval

(**Source :** Chowdhury, G.G. Introduction to Modern Information Retrieval. – London: Library Association, 1999. – p.4)

The diagram highlights the two major activities of IRS, viz., organisation of information to facilitate retrieval, and search and retrieval of information relevant to an information need. The retrieval process usually begins with a user's query expressing an information need. Expressed in a simple way, the query usually consisting of one or more search terms is matched with the index file (or with terms in the database) and the matching records are retrieved. While this is the basic process in all information retrieval activities,

in reality most IR systems and, therefore, the retrieval process, are complex in nature and involve a number of operations and tools before satisfactory and acceptable results are achieved.

Self Check Exercise

1) What do you understand by '*Information* Retrieval'? Identify the various processes involved in information retrieval.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

.....

4.3 INFORMATION RETRIEVAL VERSUS DATA RETRIEVAL

Before getting into the details of database searching, it is useful to understand the difference between '*information retrieval*' and '*data retrieval*' although many criticise this dichotomy on the grounds that the boundary between the two is not very clear. It is indeed true that the difference between the two is vague. However, when we talk of '*data retrieval*' we are normally looking for an exact match while in information retrieval this may sometimes be of interest but more generally we want to find those items which partially match the request and then select from those a few of the best matching ones. In information retrieval it is far more common to use inductive inference; relations are only specified with a degree of certainty or uncertainty and hence our confidence in the inference is variable. We can therefore characterise data retrieval as *deterministic* while information retrieval is *probabilistic*. It is probably for this reason that a separate class of software has been developed for information retrieval systems / text retrieval systems. It is therefore useful to have, at this point, a general idea of the principal differences between '*information retrieval*' and '*data retrieval*'. IRS store documents and / or their surrogates whose components vary considerably in length and character. A '*document*' is usually some kind of textual record ranging from the completely unstructured to the more structured. This is a major characteristic that is kept in mind in designing IRS. For example, WINISIS, a software package for text retrieval systems, has been used for many applications that include library collections, hospital records, database of research projects, etc. In '*data retrieval*' we are looking at systems, which are designed to manipulate highly structured data. IRS have been specifically developed to provide flexible ways of retrieving textual records. The search facilities that IRS offer are based on the premise that exact matches between search terms and field values in a record in the database are not adequate. Database searching in the context of IRS therefore assumes that sufficient degree of facilities are built in to support retrieval of records that possess a pre-defined degree of *similarity* with the query. The output of a search process in an IRS therefore is usually a list of documents each one of which has a varying degree of similarity with the query. On the other hand in '*data retrieval*' we are looking for an exact match.

Self Check Exercise

2) Identify the major characteristics that differentiate '*Information retrieval*' from '*data retrieval*'.

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

4.4 PARAMETERS FOR EVALUATION OF SEARCH OUTPUT

The two principal parameters that are widely used for evaluating the output of a search in an IRS are:

- Recall – A measure of the ability of the IRS to retrieve relevant documents; and
- Precision – A measure of the ability of the IRS to withhold irrelevant documents from being retrieved.

Users' requirements in terms of desired levels of recall and precision vary depending upon a number of factors. The search language of an IRS usually supports modification of the search strategy to regulate the recall and precision output. Three different kinds of searches have been referred to in literature:

- High Recall Search – The user wants to find all the relevant items on his query
- High Precision Search – The user wants to find only relevant items and avoid non-relevant items as far as possible
- Brief search – The user wants to find just a few relevant items. (Meadows and Cochrane, Basics of Online Searching, 1981)

4.5 SEARCH STRATEGY

Conducting an efficient and effective search requires the development and adoption of an appropriate search strategy. '*Search strategy*' is used here to mean the operations and decisions involved in the search process from the time a user approaches the system with some information need till he constructs a statement (using the syntax of the search language of the IR System) that adequately and accurately expresses the information need of the user. Obviously this involves several steps and an understanding of the issues involved in every stage is a necessary pre-requisite for carrying out effective and efficient searches.

a) **Understanding the User's Requirements:** Understanding the information requirements of the user significantly affect the search process and the output. It is important to present to the system as clear an idea of the requirements of the user as possible, before proceeding with the search process. Such clear idea is obtained as follows:

- User interview: This is a process in mediated searches through which the information intermediary seeks to obtain a clear understanding of the end-user's information requirements. The semantic gaps between the '*actual need*', '*expressed need*' and '*information need as perceived by the information intermediary*' are sought to be bridged / narrowed during this process. This is particularly important if the end user is not present during the search.

- An understanding of the requirements in terms of preferred levels of recall, precision tolerance is also necessary before proceeding with the actual search.

b) **Selecting the Database to Search:** Once an understanding of the user's requirements is obtained, the next important step is to select the database or databases to be searched. It is necessary to realise that there can be several databases that have the potential to yield relevant references. An important consideration in selecting a database is its scope. A database is not effective if it is inappropriate for the topic/user requirement. Databases vary widely in terms of their scope, coverage and timeliness. Chowdhury suggests that the following factors should be kept in mind in selecting the database to be searched:

- Subject Coverage
- Document Coverage
- Accessibility
- Period Coverage
- Search Fields
- Search Devices / Tools
- Performance, etc.

c) **Formulating the Search Statement:** It is important, before the actual search, to be clear about the topic. All possible words and / or phrases that might be used to refer to the subject should be thought of including related terms RTs, possible variations in word endings (e.g., singular, plural, adjectives); synonyms; variant terminology. In the following paragraphs an idea of how to formulate a query is provided. A query is essentially the information that a searcher supplies to the IR system in order to identify information / information resources in the system that match with the need. In practice it consists of a number of parameters that together define the boundaries of the information required. These parameters usually consist of one or more keywords/descriptors/classification codes and possibly elements defining other parameters such as dates, language, etc. The term 'keywords' will be used here to include descriptors taken from a controlled vocabulary also While a query has to be simple enough, yet it needs to be sufficiently descriptive to support retrieval of what is needed. Otherwise, the searcher will have very little control over the results. It is also highly probable that the search output would be huge necessitating screening of the retrieved texts to identify the relevant items. Unfortunately, there is no single standard method of expressing the query that exactly matches with all the relevant information available on the system and querying is largely a function of the search language / search engine that is being used. There are however, certain basic factors an understanding of which will help in formulating better queries. In its simplest form a query consists of keywords and/or phrases. When a phrase (as against a simple keyword) is input, only documents that contain the phrase will be retrieved. For example, a document that has the phrase "retrieval of information" will not match the phrase "information retrieval". Ideally, a phrase used in searching should not be too long as the chances of finding an exact match recede, as the phrase gets longer.

4.6 COMPOUND QUERIES

The basic elements of a search such as keywords and phrases (and other parameters) could be combined to constitute more powerful and effective searches. George Boole devised a system of logical operators to combine statements in symbolic form to form more complex statements. Each Boolean operator, in effect, expresses a relationship between the statements so combined. John Venn expressed these relations and thus the Boolean operators using what are now generally known as the *Venn diagrams*. Boolean operators are the most widely used devices for linking terms in searches in database. The Boolean operators ***OR***, ***AND*** and ***NOT*** correspond to the set operations: ***set union***, ***set intersection*** and ***set disjunction.*** These are diagrammatically explained in the following examples.

Example: INFORMATION OR DATA

INFORMATION AND RETRIEVAL

INFORMATION NOT DATA

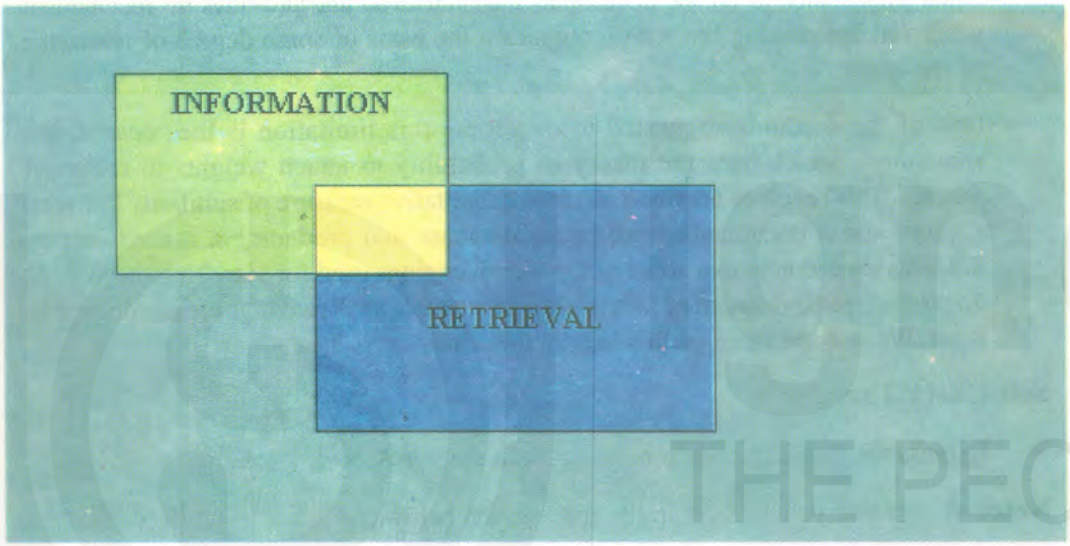


Fig. 4. 2: Database Search Using Boolean Operators

When the two search terms are linked using the Boolean **AND,** only those documents (represented by the area painted yellow) will be retrieved. The use of **OR** to link the two terms will result in the retrieval of documents corresponding to both the rectangles; the use of the search expression '**Information NOT Retrieval**' will result in retrieving only those documents represented by the area painted green. The **OR** operator is used when one is searching for documents which have either or both the terms linked this way. The use of **OR** would be appropriate to link two or more synonyms or near synonyms. The **OR** operator expands a search and treats the operands as equivalent. **AND** and **NOT** on the other hand, limit a search. One and the same search expression may also involve multiple uses of one or more of the Boolean operators. Without exception all search languages accommodate Boolean operators. In actual implementation of these, many search systems use symbols (such as *) to represent the different operators.

Example: (INFORMATION OR DATA) AND RETRIEVAL[1]

There are some major deficiencies with the Boolean operators:

a) Boolean operators merely allow retrieval of documents/web pages that contain some term or combination of terms. In practice it is often necessary to specify retrieval requirements more precisely. For example, the search for '**Information Retrieval**' has to be expressed either as

[1] It is important to note the instructions for the various search engines. Note that parentheses are used to group search terms. Also, note that the Boolean operators are represented differently using different notations on different search languages; unfortunately, no standard has been consistently followed. Many search engines are also case sensitive. Use of a singular from may not retrieve the plural form. There are also limits to the number of characters a search expression may contain.

Information AND Retrieval

or as

Retrieval AND Information

to retrieve those documents containing both **'Information'** and **'Retrieval'** rather than the required set of documents where an occurrence of the word **Information** immediately precedes the occurrence of the word **Retrieval**.

b) It is tedious to use Boolean operators to include the various morphological variants of a term (e.g., CLASSIFY OR CLASSIFICATION OR CLASSIFYING OR CLASSING)

c) Boolean operator AND is often inadequate to express the desired relation between two search terms

d) The Boolean model of searching identifies an item as relevant or not on the basis of the presence or not in the document / record of terms in the query. It attaches equal importance to all the documents thus retrieved and provides no mechanism whatever, for ranking the search output on the basis of some degree of relevance to the query.

One of the methods suggested to overcome this limitation is the '*best match searching*', which uses the theory of probability to attach weights to retrieved records. This involves adoption of some quantitative measure of similarity between a query and a document/record in the database and producing a ranked output. Students should note that some of the search engines used for searching the WWW do produce ranked output of '*hits*'. How dependable such rankings are is a debatable issue. We will see more on this later in this Unit.

Self Check Exercise

3) Express the Boolean operators using Venn diagrams

Note: i) Write your answer in the space given below.

ii) Check your answer with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

.....

.....

4.7 ADVANCED FEATURES

We have so far seen some search capabilities associated with combining keywords and phrases. Most search engines provide a number of advanced capabilities. It is, however, important to remember that all these capabilities are not available with all search engines, and some may not even provide any advanced features. In the following paragraphs we shall briefly examine some of these advanced features.

Note: i) Write your answers in the space given below.

ii) Check your answers with the answers given at the end of the Unit.

.....

.....

.....

.....

.....

.....

.....

4.12 TRENDS IN INFORMATION RETRIEVAL

The developments in computer and communication technologies have made the problems of storage and transfer of information easier. This has also made the task of searching and information retrieval more complex. In the last few decades since the advent of large online databases, efforts in IR research have been directed at overcoming some of the problems posed by this increasing complexity. There are two or three major directions in which research in this area is proceeding. While it is difficult to provide a comprehensive overview of all these areas, it is important to have a general idea of the major trends.

Intelligent IR: There have been efforts at making the computer think like a human. The development of some experimental expert systems (e.g., MYCIN) and incorporation of some features of artificial intelligence into IR systems are worth noting here. A few search engines, especially those for searching the web, are attempting to provide the facility of accepting queries in the form of a natural language query; e.g. 'What is the population of China?'. The search engine then converts the natural language query to an algorithm. There are problems with these tools and they work for some simple and/or common queries but do not work on less common types of queries. Beginning with Belkin's ASK (Anomalous State of Knowledge) hypothesis, there have also been attempts at developing better end-user models and interfaces for IR systems. User interfaces have made many advances especially in terms of elements relating to menus, forms, graphics, hypertext, etc. Natural language processing (NLP) in IR explores how natural language text (texts of documents as well as end users' queries) stored in a computer can be manipulated to transform it into a form suitable for further processing. Research in both the areas, viz,

- Designing NL interfaces that accept queries in the form of a natural language query; and

- On the storage side, NL text processing that supports structuring of large bodies of textual data to automatically derive knowledge structures (e.g., automatic classification) that may be used for accessing and retrieving information from texts is in progress. The developments appear to suggest that IR research is directed at using intelligent information processing technologies such as neural networks to overcome some of the limitations of conventional Boolean IR model. IR research in recent years has established strong links with several related areas such as NLP, Artificial Intelligence, HCI, cognitive sciences, etc. However, it is important to remember that much of this research is still in experimental stages and it will be some time before large structured commercial databases adopt these technologies.

4.13 SUMMARY

In this Unit, you have been provided an overview of the processes involved in database searching and an understanding of the various devices that are available to enhance a search. You should also have learnt the effect that the various search devices have on Recall and Precision of the search output. The most appropriate way to learn more about search devices is by using search languages and search devices in various combinations. Some of the exercises presented are aimed at this and it is hoped that the students will try using some of the search engines and search devices and obtain a first hand experience in their use.

4.14 ANSWERS TO SELF CHECK EXERCISES

1) The term '*Information Retrieval*' was coined by Calvin Mooers. It is the process of retrieving documents relevant[2] to a query from a database. In a broad sense the various processes involved in IR include both the processes associated with the creation of the database of documents as well as those associated with the search.

2) Information Retrieval Systems (also known as Text Retrieval Systems) are designed to manage texts of documents. While the record structure of IRS supports structuring data into '*fields*', it differs from DBMS which are built to manage highly structured data. For example, a relational DBMS such as INGRES manipulates highly structured and inter-related tables (relations) of data. Such DBMS may be suitable for certain library operations such as circulation, but may impose limitations in storing texts / abstracts of texts, etc., which are necessary in a text retrieval system. In terms of expected output also, there is one major difference; in data retrieval we look for records that are an exact match for the query while in information retrieval we would like to retrieve records on the basis of their degree of similarity to the query.

3) Please refer to any good textbook on Information retrieval for this

4) Search tools such as WINSPIRS and the search language of the WINISIS support a wide range of search devices including Boolean operators, Proximity operators, Truncation, Field –Specific searching, free text searching, etc. There are however limitations. For a complete understanding of these refer to the relevant manuals.

5) Boolean operators are supported by Web search engines. Many of them also support Proximity operators, Phrase searching and also provide search features relevant in the context of Web searching (e.g., link searching). Truncation, however, is not a feature widely supported by web search engines. Greg Notess provides a good overview and comparative analysis of the search features of some widely used web search engines.

(Visit SearchEngineShowdown.com; you can also get to these from the homepage of Greg Notess www.notess.com)

4.15 KEYWORDS

Boolean Operators : A system of three symbolic logical operators to combine search terms to constitute a search statement, representing '*logical sum*' (+) **(OR)**, '*logical product*' (x) **(AND)** and '*logical difference*' (-) **(NOT)**

Data : Raw numbers or letters stored in a computer which represent facts about entities

[2] Students are advised to note that the term '*Relevance*' in information retrieval is far from being clearly defined. Tefko Saracevic has discussed the many definitions of '*relevance*'.

Database : An organised collection of data that exists for the purpose of providing information

Descriptor : A word / term taken from a controlled vocabulary – usually a thesaurus – and assigned to a text to represent, partially or fully, the '***aboutness***' of the text (*see also Keyword*)

Field-Specific search : A search facility that allows the searcher to specify the field (e.g. Title field) in which a search term should be present

Information Retrieval : The process of retrieving information about documents in a database relevant to a query

Keyword : Technically, a significant word / term taken from the text of a document (*see also Descriptor*)

Proximity Operator : A set of operators to combine search terms that permit the searcher to specify the context in which a search term should occur. Proximity operators allow the searcher to specify whether two search terms should occur adjacent to one another or the maximum number of other words that could be permitted between the search terms

Search Engine : Large databases of web resources along with a search language of their own to support a wide range of search features to search the databases.

Search Language : An artificial language used to construct search statements that can be input to an IRS;

Truncation : A search device that supports searching of records containing any one or more of a number of different terms by merely specifying a string of characters that is common to the different search terms. The common string of characters could be in the beginning (root) or at the end or even in the middle of a word; widely used for words having morphological variations

4.16 REFERENCES AND FURTHER READING

Information retrieval is considered the essence of Information Science. There is therefore no dearth of good reading material and standard textbooks. **G.G. Chowdhury**'s book referred to in this Unit is a comprehensive textbook covering adequately all aspects of database searching including web searching. Reference has also been made to the website of **Greg Notess** that provides useful material especially on web searching and search engines. **F. W. Lancaster** has written standard texts on information retrieval. Besides these, there are other useful materials that students will benefit from:

Ashford, J. and Willet, P. (1988). *Text Retrieval and Document Databases.* Bromley: Chartwell-Bratt.

Bates, Marcia J. (1981). Search Techniques. *Annual Review of Information Science and Technology.* 16, 139-169.

Ingwersen, P. (1996). Interaction: Elements of a Cognitive IR Theory. *Journal of Documentation, 52*(1), 3-50.

Meadow, C.T. and Cochrane. P. A. (1981) *Basics of Online Searching*. New York: Wiley.

ignou
THE PEOPLE'S
UNIVERSITY