GCA S3 02

Database Management Systems

SEMESTER - III

BACHELOR OF COMPUTER APPLICATION

BLOCK 1

KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakati Mahanta,	Gauhati University
Prof.(Retd.) Pranhari Talukdar,	Gauhati University
Dr. Jyotiprakash Goswami,	Assam Engineering College

Course Coordinator

Dr. Sanjib Kr. Kalita,	KKHSOU
Dr. Tapashi Kashyap Das,	KKHSOU
Sruti Sruba Bharali,	KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
1, 5 & 6	Dr. Tapashi Kashyap Das, KKHSOU
2	Arabinda Saikia, KKHSOU
3 & 4	Dr. Sangeeta Kakoty, KKHSOU

Editorial Team

Content	:	Dr. Tapashi Kashyap Das, KKHSOU (Unit 2, 3 and 4) Dr. Sanjib Kr. Kalita, KKHSU (Unit 1, 5 and 6)
Language	:	Prof. (Retd.) Robin Goswami, Cotton College
Structure, Format & Graphics	:	Dr. Tapashi Kashyap Das, KKHSOU

July 2018

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 License (international): http://creativecommons.org/licenses/by-nc-sa/4.0/

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters : Patgaon, Rani Gate, Guwahati - 781017
City Office : Housefed Complex, Dispur, Guwahati-781 006; Web: www.kkhsou.in

The University acknowledges with thanks the financial support provided by the **Distance Education Bureau**, **UGC** for the preparation of this study material.

COURSE INTRODUCTION

The traditional method for storing computer data was data files. Data files have been very popular since 1960s. In fact, even today, many major computer applications run on file-based computer systems. In the last few decades, Database Management Systems has become a subject of great significance in the IT industry. In today's competitive environment, Database and Database Management Systems (DBMS) have become essential for managing our business, governments, banks, universities and every other kind of human endeavour. A database is a collection of related data organised in a way that data can be easily accessed, managed and updated. DBMS is a software package that allows data to be effectively stored, retrieved and manipulated. DBMS also provides protection and security to database. This course is on **Database Management Syatems (DBMS)**. The course is divided into two blocks :

The units in **Block 1** can be used in an introductory course on database systems. This block discusses Data Models, Keys and Relational Database design.

The units in **Block 2** mainly discusse normalization techniques, Structured Query language and database recovery as well as security.

Each unit of these blocks includes some along-side boxes to help you know some of the difficult, unseen terms. Some "EXERCISES" have been included to help you apply your own thoughts. You may find some boxes marked with: "LET US KNOW". These boxes will provide you with some interesting and relevant additional information. Again, you will get "CHECK YOUR PROGRESS" questions. These have been designed for you to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with "ANSWERS TO CHECK YOUR PROGRESS" given at the end of each unit.

BLOCK INTRODUCTION

This is the *first block* of the course '***Database Management Systems***'. After completing this block, you will be able to gain the knowledge of database and DBMS. This block consists of the following six units:

Unit - 1 introduces File Structure. Some basic concepts like data, information, field, record and files are described in this unit. This unit will help you to understand operation on files and different file organization techniques.

Unit - 2 discusses database system. Various concepts like data independence, database architecture, DBMS and its types, merits and demerits of DBMS are discussed in this unit.

Unit - 3 is on Data Models. Conceptual, physical and logical data models are discussed at the beginning of this unit. At the end, Entity-Relationship model is discussed. The concepts of entity, attributes and relationships are discussed in this unit.

Unit - 4 is on Relational model. Concepts like Entity intigrity, Referenctial intigrity are included in this unit.

Unit - 5 is on keys. Different types of keys related with database are discussed in this unit.

Unit -6 discusses the important Relational Database Design. Universal relation, Functional dependencies, Prime and Non-prime attributes are discussed in this unit.

The structure of Block 1 is as follows :

Unit –1 : File Structure

Unit –2 : Database System

Unit –3 : Data Models

Unit –4 : The Relational Model

Unit –5 : Keys

Unit –6 : Relational Database Design

BACHELOR OF COMPUTER APPLICATION

Database Management Systems

DETAILED SYLLABUS

BLOCK-1
Semester 3

Page No.

UNIT 1: File Structure 7-18
Data and Information, Concept of Field, Key field; Records and its types, Fixed length, records and Variable length records; Files, Operation on files, Primary file organization

UNIT 2: Database System 19-45
Traditional file approach versus Database approach; Data Independence, Database System, Database Architecture, The Three levels of Architecture, Mapping, Database Administrator, The Database Management System, Types of DBMS, Merits and demerits of DBMS.

UNIT – 3: Data Models 46-66
Conceptual model, Logical model, Physical model, ER model as a tool for conceptual design: Entities, Attributes and Relationships, Weak and Strong entities, Conversion of ER model into Relational schema, ER modeling symbols.

UNIT – 4: The Relational Model 67-76
Relational data model concepts, Integrity constraints: Entity integrity, Referential integrity, Domain Constraints.

UNIT – 5: Keys 77-91
Concept of keys, Composite key, Candidate key, Primary key, Alternate key, Foreign key, Defining Primary and Foreign keys in Database.

UNIT – 6: Relational Database Design 92-107
Database Design, Decomposition, Universal Relation, Functional dependencies, Prime and Non-prime attributes.

UNIT 1: FILE STRUCTURE

UNIT STRUCTURE

1.1 Learning Objectives
1.2 Introduction
1.3 Data and Information
1.4 Fields and Records
1.5 Files
    1.5.1 Operation on Files
1.6 Primary File Organization
    1.6.1 Sequential Access Organization
    1.6.2 Direct Access Organization
    1.6.3 Indexed-Sequential Access Organization
1.7 Let Us Sum Up
1.8 Answers to Check Your Progress
1.9 Further Reading
1.10 Model Questions

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define Data and Information
- define Fields, Records and Files
- describe operation on Files
- describe different File organization techniques like Sequential access, Direct access, Index-Sequential access etc.

1.2 INTRODUCTION

Computer files and file systems have great similarities with traditional files and file systems. All basic operations like insertion, deletion, updation, searching, etc. are possible in both the systems. For the formation of computerised record keeping systems we are required to introduce Database and Database Management Systems (DBMS).

Some basic terms like data, information, fields, records, files etc. and their organization are prerequisite to understand database and DBMS.

This unit is an introductory unit and gives you an understanding of those basic terms and concepts.

1.3 DATA AND INFORMATION

Data and its efficient management have become an important issue with the growing use of information technology and computers. Proper organization and management of data is necessary to run the organization efficiently.

Data are raw or isolated facts from which required information can be produced. There are uncountable examples of data such as name of a person, name of a place, the roll number of a student, the ticket number of a passenger, passport number of a person, PAN card number, credit card number, identification number for an employee etc.

Data are processed to generate information which is meaningful for the recipient. Information helps human beings in their decision making process. Some common examples of information are: time table of a class , report card, printed documents, pay slips, invoices, bank statements, receipts, reports etc. The information is obtained by assembling items of data into a meaningful form. For example, marks obtained by students and their roll numbers are *data*, but the the report card/mark sheet generated from these data are *information*. Thus, ***information*** may be defined as the collection of related data that when put together gives meaningful and usefull message to a recipient who uses it for different purposes like decision making. Data and infomation are closely related and are often used interchangeably.

1.4 FIELDS AND RECORDS

Data item or Fields

A data item or field is the smallest unit of information that has meaning to its user. For example, in a Telephone bill, *Name*, *Telephone_no*, *Bill_amount*, *Address* are Fields. Similarly, in an employee salary slip, basic

pay, gross pay, HRA, DA, deduction etc. are examples of Fields.

Records

A ***record*** is a collection of logically related fields. Each record contains unique and uniform information that is divided into fields. This uniformity allows for consistent access of information. An example of a student record is shown in figure 1.1. A record consists of values for each field.

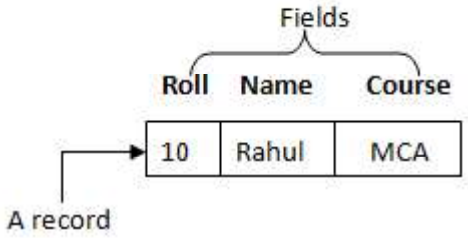


Figure.1.1: An example of record

A file can contain ***fixed length record*** and ***variable length record***. Each field in a *fixed length record* is given a fixed length. The length of the record should be large enough to hold even the largest value anticipated for that field. In a student's record we may provide 15 spaces for the name field, 3 for roll number, 2 for each percentage field, 4 for division etc. Here the length of each field is the same for all records irrespective of the values contained in them. It may result in wastage of storage space in the file if all names are not of 15 characters.

In case of *variable length record*, the size of the data item are not fixed. The value in every record is allowed to take up as much space as is required by its size which enables us to save disk space. The end of each item is recognized by a delimiter such as a comma, or a colon. The following example will make the definition clear. The gap between each data item is having a space which is counted as a character.

4, Karabi Roy, 72, 1st (22 characters)

In the above record, the respective fields are Roll_No, Name, Percentage and Disvision. This record occupies 19 characters including the blank spaces and commas.

When we use fixed-length records, we need to make the record length equal to the length of the longest record. If we need to store many short records with occasional long ones, using fixed-length records wastes a lot of disk space, so variable-length records would be a better choice.

1.5 FILES

A file is a collection of related sequence of records. Records with identical formats/types are generally kept in a file. If every record in the file has exactly the same size in bytes, the file is said to be made up of fixed-length records. If the records in the file are of different sizes, the file is said to be made up of *variable-length records*.

For example, in Figure 1.2, a student result file is shown in tabular format. It contains ten records of students and each record has four related fields namely ***Roll_No***, ***Name***, ***Percentage***, ***Division***. In the file, the information about a particular student in one line or row of the table is an example of record. The collection of result information for all the students (all columns and rows), i.e., the entire table is an example of ***file***.

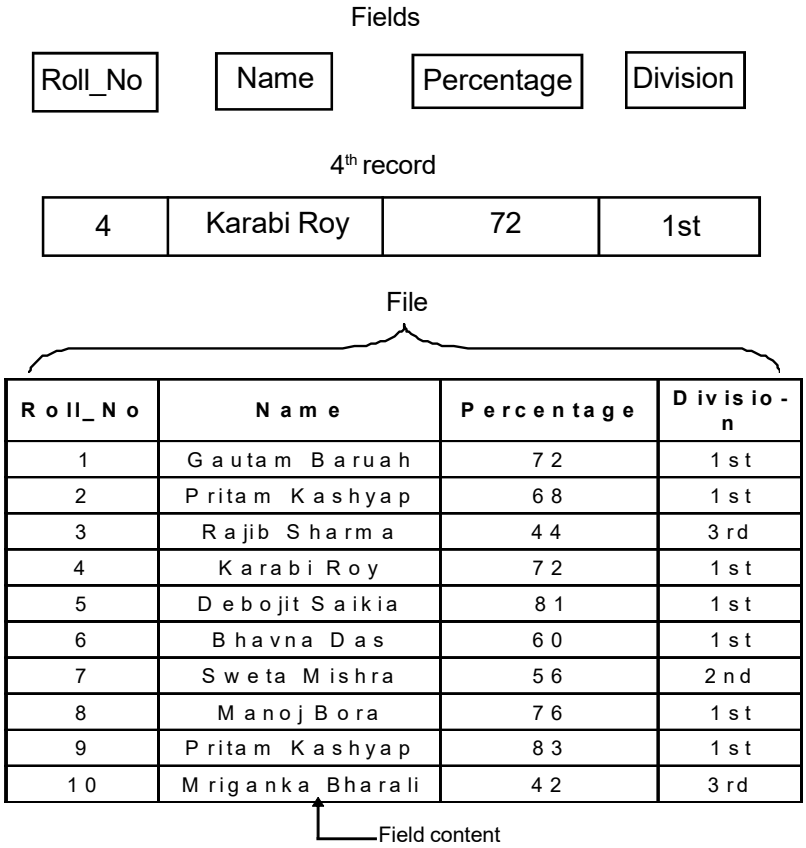


Fig. 1.2: Concept of field, records and files

Each record in a file may contain many fields, but the value in a certain field may uniquely determine the record in the file. Such a field is known as ***key field***. In case of student's record in Figure 1.2, *Roll_No* is the key

field because it is unique for each student. Similarly, *Part_no* in a stock file, *Account_no* in a bank customer's file are all examples of key fields.

A file can be of two types:

Master File and ***Transaction File***

A *master file* contains records of relatively permanent data. For example, the name, roll number, sex, date of birth of a student would appear in a student master file as these records are.

A *transaction file* is a file in which day to day operational data are stored. Transaction file contains the records that are used to update the records of the master file. For example, a transaction file may contain the information of students who pay their monthly fees. This file will update the record of the student whose roll number matches with the transaction file record. The record of the master file will thus be updated as and when necessary.

1.5.1 Operation on Files

There are various operations associted with files. Let us take a real life example where files are being maintained to keep records. In a school, class wise student files are maintained where information about each student is recorded and all these records are placed together within a file cover with a name on it. For different classes different student files are normally maintained. Again, all such files can be stored together in a shelf or a file-cabinet (like a folder in a computer). To retrieve any information about a particular student of a specified class, the concerned class file (identified by the filename) can be taken out and the records within that file can be searched out to get the details of the student. If the file records remain sorted with respect to roll numbers, then the roll numbers of the students can be treated as a search key.

A computer can also work in the same way. Computer files also facilitate easy storage, retrieval, and manipulation of data. Instead of paper files, computer store electronic files in a hard disk or removable

disk in the form of bits and bytes. Therefore, a disk file is nothing but a collection of records. The record can be entered through a keyboard and saved as a file in the hard disk. Various operations are associated with computerised files. There are three major operations on files as given below :

(a) **Insertion of new records**:

This is the process of adding data or record in a file at the indicated location. A record is inserted in a sequential file at the end of the file.

(b) **Modifications of some existing records**:

This operation is done to modify old data or record with a new data or record in a file at the indicated location.

(c) **Deletion of records**:

This is the process of removal of records or data at the specified location.

Maintaining records and files in proper format is very essential. It enables the users to access information easily and quickly. In any organization the maintenance is basically carried out by the system administrator of the system. It involves integrity checking, missing of null value checking, corrupted data checking etc. We will discuss the concept of data integrity and null value later in this block.

1.6 PRIMARY FILE ORGANIZATION

A file containing records may have the organization depending upon the way these are arranged in the file. A file is organized to ensure that records are available for processing. Following are the tree types of file organization:

- Sequential Access organization.
- Direct Access organization.
- Indexed-Sequential Access organization.

1.6.1 Sequential Access Organization

In case of sequential file, records are arranged in some order. For example, a student's file may be kept in ascending order of roll numbers. It is not necessary that the records of a sequential file should be physically in adjacent positions. However, on a magnetic tape for sequential organizations, the records are written one after the other along the length of the tape. In case of disks, the records of a sequential file may not be in contiguous locations. The sequential order may be maintained with the help of pointer in each record. To access a record, previous records within the block are needed to be scanned. Thus, sequential record design is suitable for reading one record after another without a search delay. In a sequential organization, records can be added only at the end of the file. It is not possible to insert a record in the middle of the file without rewriting the file. However, in a database system a record may be inserted anywhere in the file, which would automatically re-sequence the records following the inserted record. Another approach is to add all new records at the end of the file and later sort the file on a key (name, number, etc). Information on a sequential-access device can only be retrieved in the same sequence in which it is sorted.

Sequential processing is quite suitable for such applications like preparation of monthly pay slips, or monthly electricity bills etc., where most, if not all, of the data records need to be processed one after another. In these applications, data records for every employee or customer needs to be processed at scheduled intervals (in this case monthly). However, while working with a sequential-access device, if an address is required out of order, it can only be reached by searching through all those addresses, which are stored before it. For instance, data stored at the last locations cannot be accessed, until all preceding locations in the sequence have been traversed. This is analogous to a music tape cassette. Suppose you like to hear a particular song which is in 8th position in the cassette. For that you can "fast forward" the first seven songs. Although, not fully played, the 7 songs are still accessed. Magnetic tape is an example of sequential access storage

device. Following are the main drawbacks and advantages of sequential file organization:

Drawbacks :

- Sequential files are not suitable for on-line enquiry where up-to-date information is required.
- Information on the file is not always current.
- Addition and deletion of records are not a simple task.
- Searching information in a sequential file can be a very slow process. For any search operation, we need to start reading a sequential file from the beginning and continue till the end, or untill the desired record is found, whichever is earlier. This is both time-consuming and cumbersome.

Advantages:

- File design is simple.
- Low-cost file medium. Tape can be used.

1.6.2 Direct Access Organization

There is a popular type of file, called *direct files* which permit random access or direct access. In case of direct file organization records are placed randomly throughout the file. Records need not be in sequence because they are updated directly and rewritten back in the same location. New records are added at the end of the file or inserted in specific locations based on software commands. Records are accessed by addresses that specify their disk locations. An address is required for locating a record, for linking records, or for establishing relationships. Addresses are of two types: *absolute* and *relative*.

An ***absolute*** address represents the physical location of the record. It is usually stated in the format of sector/track/record number. For example, *3/16/4* means *go to sector 3*, *track 16 of that sector*, and the *fourth record of the track*. One problem with absolute address is that they become invalid when the file that contains the records is relocated on the disk.

A ***relative*** address gives record locations relative to beginning of the file. There must be fixed-length records for reference. Another way of locating a record is by the number of bytes it is from the beginning of the file.

1.6.3 Indexed-Sequential Access Organization

An *indexed-sequential* file is basically a file organized serially on a key field. In addition, an *index* is maintained which speeds up the access of isolated records. The index provides random access to records, while the sequential nature of the file provides easy access to the subsequent records as well as sequential processing. Indexed-sequential access organization reduces the magnitude of the sequential search and provides quick access for sequential and direct processing. The primary drawback is the extra storage space required for the index. It also takes longer to search the index for data access or retrievals. The retrieval of a record from a sequential file, on average, requires access to half the records in the file. To improve the query response time of a sequential file, a type of indexing technique can be added. The purpose of indexing is to expedite the search process. Indexes created from a sequential set of primary keys are referred to as indexed-sequential. Just as we use the index to locate information in a book, an index is provided for the file. Some advantages and disadvantages of this organization are given below:



Index:
An index is a table of records arranged in a particular fashion for quick access to data.

Advantages:

- Up-to-date information will always be available on the file
- It is suitable for on-line or derect access processing.

Disadvantages:

- Less efficient in the use of storage space.
- Relatively an expensive medium.



CHECK YOUR PROGRESS

1. State whether the following statements are true (T) or false (F) :
   (i) A record is inserted in a sequential file at the end of the file.
   (ii) Information is data that has been processed into a more useful form.
   (iii) A collection of fields constitutes a file.
   (iv) A file needed for updating a master file is called transaction file.
   (v) Records of transaction files are permanent in nature.
   (vi) Magnetic tape is an example of sequential access storage device.
   (vii) In direct access file organization records are placed randomly.
   (viii) An index is maintained which speeds up the access of isolated records in case of sequential file organization.
   (ix) The smallest piece of meaningful information is called data item.
   (x) Sequential files are suitated for on-line enquiry where up-to-date information is required.



1.7 LET US SUM UP

- Data may be defined as a known fact that can be recorded and that have implicit meaning.
- Information is processed and organised data. It can be defined as collection of related data that, when put together, communicate meaningful and useful message to a recipient who uses it.
- The smallest piece of meaningful information is called a field.
- A record is collection of field values or data items of a given entity.

- A file is organized to ensure that records are available for processing. There are three types of organization used in computer to store records. They are: Sequential access, Direct access, and Index-sequential access organization.
  (a) Sequential organization means storing records in contiguous blocks according to a key field. Magnetic tape is an example of sequential access storage device.
  (b) Indexed-sequential organization stores records sequentially but uses an index to locate records. Records are related through chaining using pointers.
  (c) Direct-access organization has records placed randomly through-out the file. Records are updated directly and independently of other records.
- In direct addressing two types of addressing may be used: absolute and relative address.
- An absolute address represents the physical location of the record. It is usually stated in the formal of sector/track/record number. A relative address gives record locations relative to beginning of the file. There must be fixed-length records for reference.
- A sequential that is indexed is called an indexed-sequential file.



1.8 ANSWERS TO CHECK YOUR PROGRESS

1. (i) True (ii) True (iii) False (iv) True (v) False
   (vi) True (vii) True (viii) False (ix) True (x) False



1.9 FURTHER READING

1. Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems. Pearson.

2. Date, C. J. (2006). An introduction to database systems. Pearson Education India.



1.9 MODEL QUESTIONS

Q1. Explain Data and Information?

Q2. Explain the following terms with example: Field, Record, File.

Q3. What are the different types of file organization techniques?

Q4. Write short notes on :

(a) Direct access organization

(b) Sequential access organization

(c) Index-Sequential organization

Q5. What are the advantages and disadvantages of sequential access organization.

Q6. What are the advantages and disadvantages of indexed-sequential access organization.

*****

UNIT 2 : DATABASE SYSTEM

UNIT STRUCTURE

2.1 Learning Objectives
2.2 Introduction
2.3 Traditional File Approach
2.4 Database Approach
    2.4.1 Advantages of Database
2.5 Database Management System
    2.5.1 Merits and Demerits of DBMS
2.6 Database Architecture
2.7 Data Independence
    2.7.1 Logical Data Independence
    2.7.2 Physical Data Independence
2.8 DBMS Language
2.9 Types of DBMS
    2.9.1 Centralised DBMS
    2.9.2 Parallel DBMS
    2.9.3 Distributed DBMS
    2.9.4 Client-Server DBMS
2.10 Database Administrator
2.11 Let Us Sum Up
2.12 Further Reading
2.13 Answers to Check Your Progress
2.14 Model Questions

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define a database and a Database Management System (DBMS)
- describe DBMS architecture
- illustrate data independence and data dictionary
- explain DBMS language

- identify the components of database system environment
- describe the role of Database Administrator (DBA)

2.2 INTRODUCTION

In the previous unit, we have learnt the basic idea of data, information, fields and records. In addition, concepts of files and basic file organization techniques are also discussed. All these are the elementary concepts relating to the database system. A *database system* is a tool that simplifies the managing of data and provides necessary manipulation on data, on the basis of the users demand. It is the central repository of data in the organization's information system. It provides an array of features which can be used to ensure optimal utilization of data for enhancing effective decision making in an organization.

In this unit, we will learn a some basic concept of database and DBMS. We will also discuss the ANSI/SPARC three-tier database architecture and different types of DBMS. Besides, we will discuss the responsibility of a database administrator.

2.3 TRADITIONAL FILE APPROACH

In earlier days, an organization's information was stored as group of records in separate files. These file processing systems consisted of a few data files and many application programs as shown in the figure below. Each file, called a *flat file,* contained and processed information for one specific function, such as, accounting or inventory. At that time programmers used programming languages such as COBOL to write application programs that can directly access flat files to perform data management services and provide information for the users. Each application files and programs were created and maintained independent of other applications.

For example, if we consider the students data in a universitiy then

– student address may be needed for *the* applications like *registering, library management, financial office, grade reporting* etc.

– each application separately maintains its data files and programs to manipulate those files
– possibly the same data (e.g., length of names, address etc.) are stored in different formats in the above applications
– whenever some information regarding a group of students is updated in one application that updation may not be done simultaneously in the different applications where students records are stored.

As a result, the system will provide wrong information about the students. In addition, potentially different values and/or different formats for the same data are stored in different files which lead to not only wastage of space but also cause the redundancy.

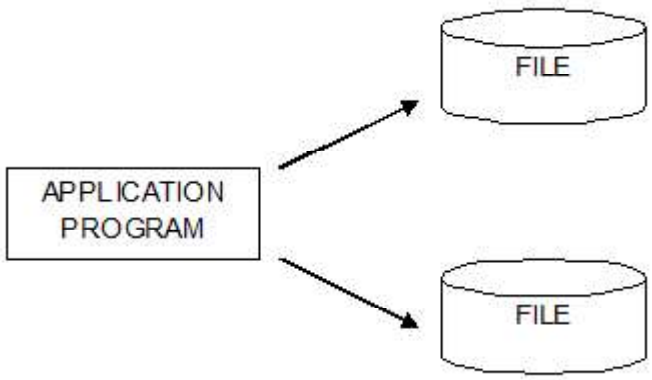


Figure 2.1: Traditional File Approach

On creating the files and programs for the file oriented system, the developers focused on business processes, or how business was transacted, and their interaction. However, business processes are dynamic, requiring continuous changes in files and applications. Moreover, programmers design the codes in accordance with the physical storage structure of data and access procedures also depends on it. Therefore, any physical changes result in the programmers rewriting the code to adjust the change.

The file-based approaches, which came into being as the first commercial applications of computers, suffered from the following significant disadvantages :

Data redundancy :

In a file system if an information is needed by two distinct applications, then it may be stored in two or more files. Repetition of the same data item in more than one file is known as *data redundancy*. This leads to an increase in the cost of data entry and data storage.

Data integrity problem :

Data redundancy also leads to data inconsistency or loss of data integrity. Data integrity refers to the consistency of data in all files. That is, any change in a data item must be carried out in every file containing that field for consistency.

Lack of data independence :

In file processing systems, files and records are described by specific physical formats that are coded into the application program by programmer. If the format of a certain record is changed, the code in each file containing that format must be updated.

Poor data control :

A file-oriented system is decentralised in nature. It means there was no centralised control at the data element level.

Incompatible file formats :

As the structure of files is embedded in the application programs, the structures are dependent on the application programming language. For example, the structure of a file generated by a COBOL program may be different from the structure of a file generated by a 'C' program. The direct incompatibility of such files makes them difficult to process jointly.

2.4 DATABASE APPROACH

An alternative approach to the traditional file processing system is the modern concept, known as the database approach. A database is an organised collection of records and files which are related to each other. In

a database system, a common pool of data can be shared by a number of applications as it is data and program independant. Thus, unlike a file processing system, data redundancy and data inconsistency in the database system approach are minimised. In this approach, the user is free from the detailed and complicated task of keeping up with the physical structure of data. A clelar-cut distinction between traditional file system and database system is depicted by the following diagram.

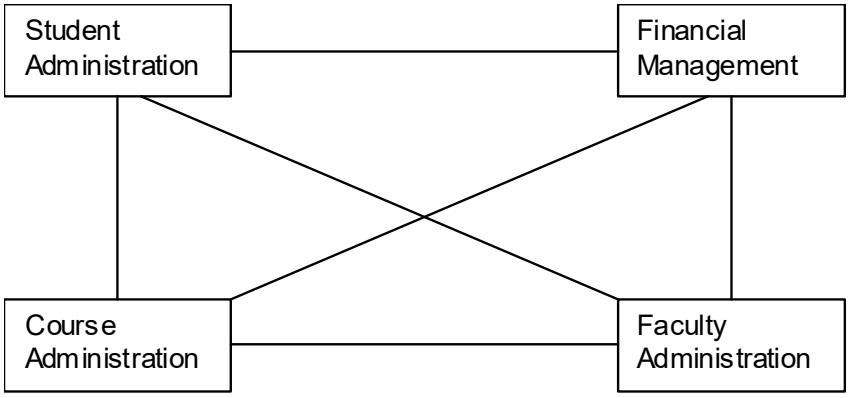


Figure 2.2: Traditional file approach

Here in this figure, the traditional record keeping system in a university is shown, where all applications are interrelated and cause repetation of the same data in different files leading to the problem of data redundancy and inconsistency.

In the figure below, indicates how several applications share common data in a database approach.

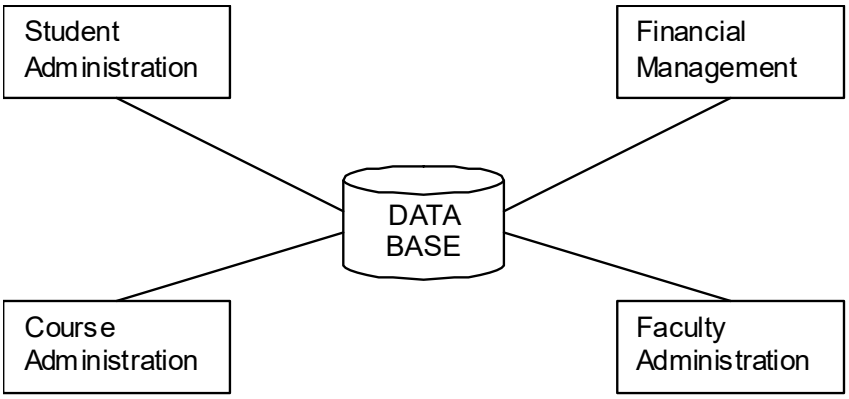


Figure 2.3: Database approach

Always remember - a database is organised in such a way that a computer program can quickly select the desired piece of data. A database can further be defined as under. It -

- is a collection of interrelated data stored together without harmflul or unnecessakry redundancy.
- stores data indepedent of programs, and any changes in data storage structure or access strategy donot require changes in accessiing programs or queries.
- serves multiple applications in which each user has its own view of data. The data is protected from unauthorised access by security mechanism and concurrent access to data is provided with recovery mechanism.

Broadly, the objectives of the database approach are to make informaltion access easy, fast, relatively inexpensive and flexible for the user. The specific objectives may be listed as follows :

- controlled data redundancy
- enhanced data consistency
- data independence
- application independence
- ease of use
- econommical, and
- recovery from failure

2.4.1 Advantages of Database

Database approach provides the following benefits over the traditional file processing system :

Redundancy control :

In a file processing system, each application has its own data, which causes duplication of common data item in more than one file. This data duplication needs more storage space as well as multiple updation for a single transaction. This problem is overcome iin database approach where data is stored only once.

Data consistency :

The problem of updating multiple files in file processing system leads to inaccurate data as different files may contain different information of the same data item at a given point of time. In database approach, this problem of inconsistent data is automatically solved wiith the control of redundancy.

Thus, in a database, data accuracy or integrity or accessibility of data is enhanced to a great extent.

Data Independence :

This means that data and programs are independent. Most of the file processing systems are data dependent, which implies that the file structures and accessing programs are interrelated. However, the database approach provides an independence between the file structure and program structure.

Sharing of data and security :

Data in a database are shared among the users and the applications. In database approach data are protected from unauthorised access.

2.5 DATABASE MANAGEMENT SYSTEM

The *database management system* is the interface between the users (application programmers) and the database(the data). A database management system(DBMS) is a program that allows the user to define, manipulalte and process the data in a database, in order to produce meaningful information.

A DBMS is a set of *software programs* that controls the *organization, storage, management, and retrieval of data* in a database. It is a set of pre-written programs that are used to store, update and retrieve a database. The DBMS accepts requests for data from the application program and instructs the *operating system* to transfer the appropriate data. The follow-ings are the examples of DBMS software :

Microsoft Visual FoxPro, MonetDB, MySQL , Oracle Database, PostgreSQL, SQL Anywhere, SQLite, FileMaker, IBM DB2, IBM UniVerse, Firebird, Microsoft Access, Microsoft SQL Server etc.

The following are examples of database applications :

- reservation systems, banking systems
- record/book keeping (corporate, university, medical), statistics
- bioinformatics, e.g., gene databases
- criminal justice
  - fingerprint matching
- multimedia systems
  - image/audio/video retrieval
- satellite imaging; require petabytes (10^{15} bytes) of storage
- the web
  - almost all data-intensive websites are database-driven;
- data mining (Knowledge Discovery in Databases) etc.

To complete our initial definitions, we will call the database and DBMS software together as a *database system*. Figure 2.4 depicts a database system.

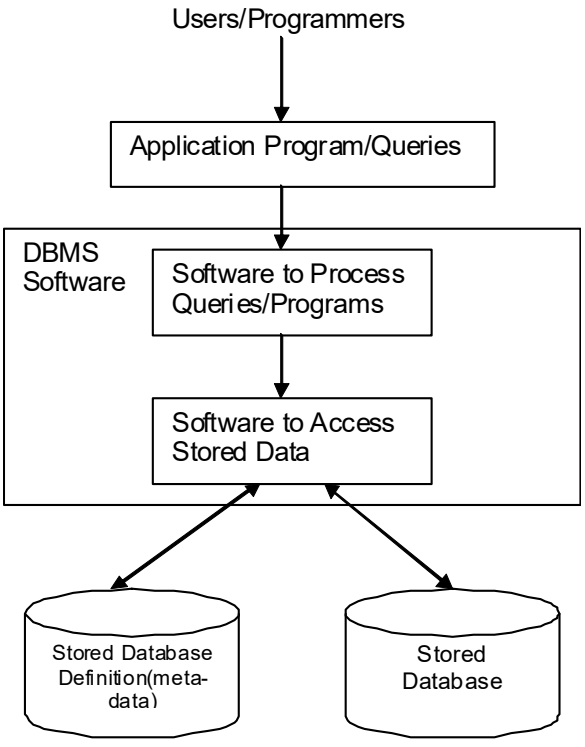


Figure: 2.4: A simplified database system

2.5.1 Merits and Demerits of DBMS

Due to the centralised management and control, the database management system has numerous advantages, some of which are explained below :

a) Merits :

i) Minimal data redundancy :

Centralized control of data avoids the unnecessary duplication of the data and effectively reduces the total amount of data storage required. It also eliminates the extra processing necessary to trace the required data in a large storage of data. Another advantage of avoiding duplication is the elimination of the inconsistencies that tend to be present in redundant data files.

ii) Program-data independence :

The separation of metadata(data description) from the application programs that use the data is called data independence. In the database environment, it allows for changes at one level of the database without affecting the other levels. With the database approach, metadata are stored in a central location called repository. This property of data systems allows an organizations data to change and develop without changing the application programs that process the data.

iii) Efficient data access :

DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

iv) Improved data sharing :

Since, database system is a centralised repository of the data belonging to the entire organization(all departments), it can be shared by all authorised users. Existing application programs as well as new application programs(which are designed on the basis of the existing data) can share the data in the database.

v) Data Integrity :

Integrity of data means that the data in a database is always accurate, such that incorrect information cannot be stored in a database. In order to maintain the integrity of data, some integrity constraints are enforced on the database.

vi) Data security :

Database security is the protection of database from unauthorised users. The DBA (Database Administrator), can define security rules to check unauthorised access to data. Some users may be given rights to retrieve the data only, whereas others may be permitted to retrieve and edit the data. The DBA can formulate different rules for each type of access (retrieve, modify, delete, etc) to each piece of information in the database.

vii) Enforcement of standards :

With the central control of the database, a DBA can define and enforce the necessary standards. Applicable standards might include any or all of the following : departmental, organizational, industry, corporate, national or international. Standards can be defined for data formats to facilitate the exchange of data between systems, naming conventions, display formats, terminology, report structure etc. The data repository provides DBAs with a powerful set of tools for developing and enforcing these standards.

viii) Providing Backup and Recovery :

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

b) Demerits :

The **demerits** of the database approach are summarized below :

i) Complexity:

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

ii) Size :

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

iii) Performance :

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

iv) Higher impact of a failure :

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

v) Cost of DBMS :

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

vi) Additional Hardware cost :

The disk storage requirements for the DBMS and the database necessitate the purchase of additional storage space. Furthermore,

to achieve the required performance it may be necessary to purchase a large machine, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

vii) Cost of Conversion:

The cost of the DBMS and extra hardware may be significant compared to the cost of converting existing applications to run on the new DBMS and hardware. This cost also includes the cost of training staff to use these new systems and possibly the employment of specialist staff to help with conversion and running of the system. This cost is one of the main reasons why some organizations feel tied to their current systems and cannot switch to modern database technology.



CHECK YOUR PROGRESS

1. Select the correct answer :

(a) Which is not a DBMS packages?

(i) Unify (ii) Ingress

(iii) IDMS (iv) All are DBMS packages

(b) Find the wrong statement

Database software

(i) provides facilities to create, use and maintain database.

(ii) supports report generation, statistical output, graphical output.

(iii) provides routine for backup and recovery.

(iv) all are correct.

(c) Which one of the following is not a valid relational database?

(i) SYBASE (ii) IMS

(iii) ORACLE (iv) UNIFY

(d) Centralized control is

(i) advantage of a DBMS (ii) disadvantage of a DBMS

(iii) Both (i) and (ii) (iv) None of the above

(e) Data are

(i) Raw facts and figures (ii) Information

(iii) Electronic representation of facts

(iv) None of these

2.6 DATABASE ARCHITECTURE

So far, we have come to know that, a DBMS is a collection of interrelated files and a set of programs that allow several users to access and modify these files. A major purpose of a database system is to provide the users with an abstract view of the data. That is the system hides certain details of how the data is stored and maintained. We can imagine that the whole database system is divided into levels. The generalised architecture of a database system is called the ANSI/SPARC (American National Standards Institute - Standards Planning and Requirements Committee) model.

ANSI/SPARC three-tier database architecture is shown in the Fig. 2.5

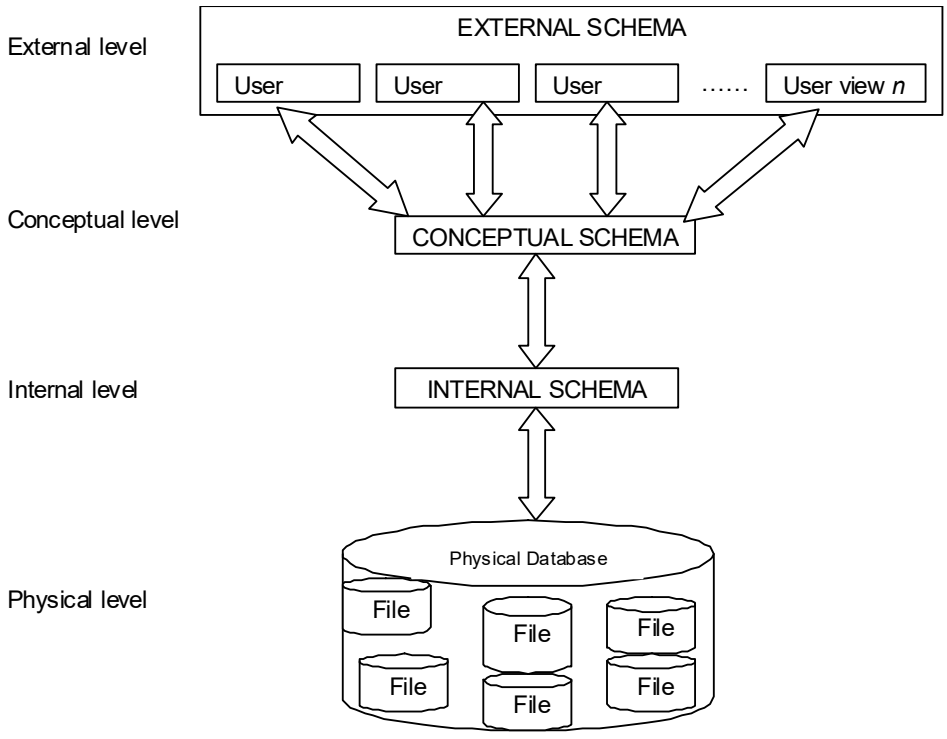


Fig.2.5: Three-tier database architecture

It consists of the following three levels :

- External level or view level,
- Conceptual level,
- Internal level or physical level.

External level :

The external level is the user's view of the database and closest to the users. This level describes that part of the database that is relevant to the user. Most of the users of database are not concerned with all the information contained in the database. Instead, they need only a part of the database relevant to them. For example, even though the bank database stores a lot more information, an *account holder* would be interested only in the account details such as the current balance and the transactions made. They may not need the rest of the information stored in the account holders database. An e*xternal schema* describes each external view. The external schema consists of the definition of the logical records and the relationships in the external view.

In the external level, the different views may have different representations of the same data. The figure below (Fig. 2.6) describes the different views of the database related to different users.

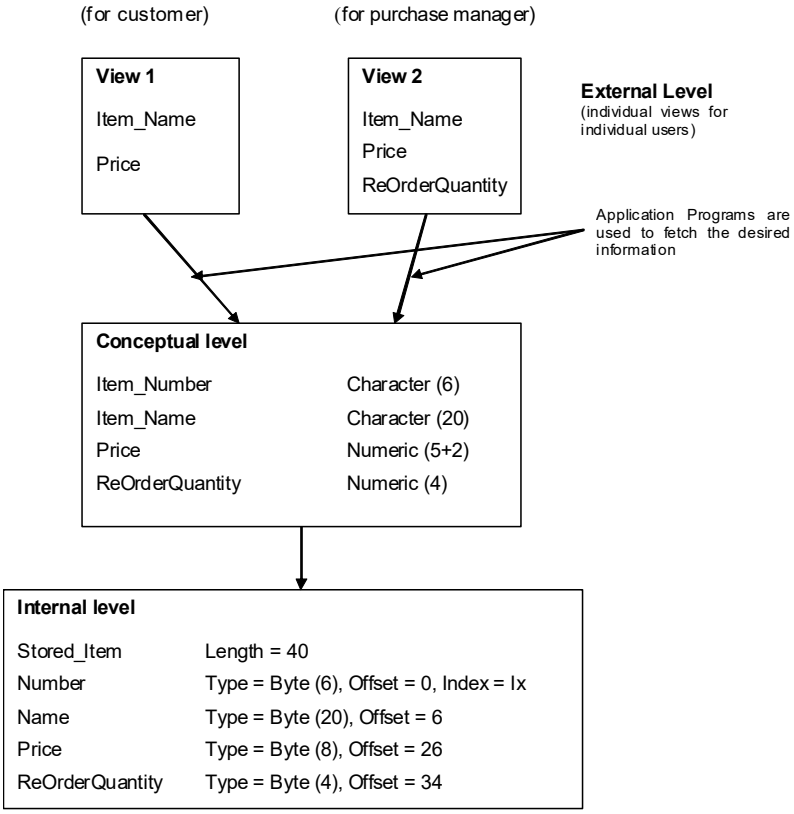


Fig.2.6: View of data at three-tier database architecture

Conceptual level :

Conceptual level is the middle level of the three-tier architecture. At this level of database abstraction, all the database entities and relationships among them are included. Conceptual level provides the community view of the database and describes what data is stored in the database and the relationships among the data. One conceptual view represents the entire database of an organization. It is a complete view of the data requirements of the organization that is independent of any storage consideration. The *conceptual schema* defines conceptual view. It is also called the logical schema. There is only one conceptual schema per database.

Internal level or physical level :

The lowest level of abstraction is the internal level. It is the one closest to the physical storage device. This level is also termed as physical level, because it describes how data are actually stored on the storage medium such as hard disk, magnetic tape etc. This level indicates how the data will be stored in the database and it also describes the data structures, file structures and access methods to be used by the database. The *internal schema* defines the internal level. The internal schema contains the definition of the stored record, the methods of representing the data fields and accessed methods used. The figure shows the internal view record of a database.

2.7 DATA INDEPENDENCE

Data independence is the characteristic of a database system to change the schema at one level without having to change the schema at the next higher level. This characteristic of DBMS insulates the application programs from changing the data. The data independence is achieved by DBMS through the use of the three-tier architecture of data abstraction. There are two types of data independence -

(i) Logical data independence

(ii) Physical data independence

2.7.1 Logical Data Independence

Logical data independence is the ability to change the conceptual schema without having to change the external schema or application programs. We may change the conceptual schema to expand the database(by adding a record type or data item) or to reduce the database(by removing a record type or data item). Only the view definition and the mapping need to be changed in a DBMS that supports logical data independence. After a logical change in the conceptual schema, the application programs that refer to the external schema construct must work as before.

2.7.2 Physical Data Independence

Physical data independence implies the ability to change the internal schema without changinig the conceptual(or external) schemas. Changes to the internal schema may be required for improving the performance of the retrieval or updation operations. In other words, physical data independence indicates that the physical storage structures or devices used for storing the data could be changed without changing the conceptual view or any of the external views.

2.8 DBMS LANGUAGE

A DBMS must provide appropriate languages and interfaces for each category of users to express database queries and updates. After completing the design of a data base, a DBMS is chosen to implement the database. It is important to specify first the conceptual and internal schemas for the database. Following laguages are used for specifying database schemas :

i. Data definition language (DDL)
ii. Storage definition language (SDL)
iii. View definition language (VDL)
iv. Data manipulation language (DML)

i) Data definition language (DDL)

DDL is a special language which species the database conceptual schema using set of definitions. DDL allows the DBA or the user to describe and name the entities, attributes and relationships required for the application, together with any associated integrity and security constraints. The DBMS has a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs.

For example, look at the following DDL statements :

```
CREATE TABLE EMPLOYEE
    (
    Fname          varchar(50     NOT NULL,
    Lastname       varchar(50)    NOT NULL,
    Eno            char(9)        NOT NULL,
    DOB            date,
    Address        varchar(60),

    PRIMARY KEY (Eno),
    );
```

The execution of the above DDL statements will create an EMPLOYEE table as shown below :

EMPLOYEE				
Fname	Lastname	Eno	DOB	Address

ii) Storage definition language (SDL)

Storage definition language is used to specify the internal schema in the database. In SDL, the storage structure and access memthods used by the database system is specified by set of statements.

iii) View definition language (VDL)

View definition language is used to specify user's views(external schema) and their mappings to the conceptual schema. There are two views of data - *logical view (*refers to the programmers view) and *physical view* (reflects the way how the data are stored on disk).

iv) Data manipulation language (DML)

DML provides a set of operations to support the basic data manipulation operations on data in a database. Data manipulation is applied to all the three(conceptual, internal, external)l levels of schema. The part of DML that provides data retrieval is called *query language*. DML provides the following data manipulation operations on a database :

- retrive data or records from database
- insert (or add) records to database
- delete records from database
- retrieve records sequentially in the key sequence
- retrieve records in the physically recorded sequence
- retrieve records that have been updated
- modify data or record in the database file

In other words, we can say that DML helps in communicating with the DBMS.

2.9 TYPES OF DBMS

The modern business environment revolves around the accuracy and integrity of information. The advancements in computer technology and rapid development of graphical user interface (GUI)-based applications, networking and communications have resulted in new dimensions in database management systems. A DBMS can be classified according to the number of users, the database site locations and the type and extent of use.

On the basis of number of users **:**

- Single-user DBMS
- Multi-user DBMS

Onthe basis of site locations :

- Centralised DBMS
- Parallel DBMS
- Distributed DBMS
- Client/Server DBMS

On the basis of the type and extent of use :

- Transactional DBMS
- Decision support DBMS
- Data Warehouse

We will mainly discass centralised, parallel, distaibuted and client/server DBMS in the following section.

2.9.1 Centralised DBMS

The centralised database system consists of a single computer system associated with its peripherals, physically located in a single location. The computer system offers data processing facilities to the users located either at the same site, or, at geographically dispersed sites, through remote terminials. The management of the system and its data are controlled centrally from any one or central site. The following figure shows a centralised database system.

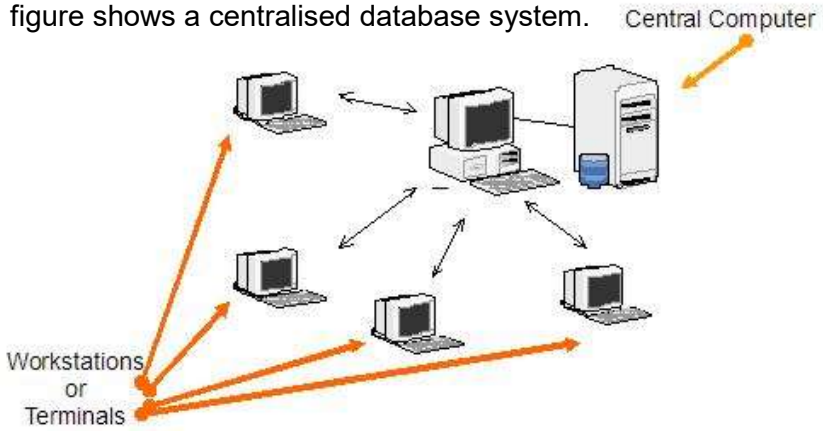


Fig.2.7: Centralised DBMS

Advantage of such sytem are given below :

- Most of the functions such as update, backup, query, control access etc. are easier with this system.

- Single database is shared accross the several different users.

Ofcourse, when the central site computer goes down, then every user is blocked from using the system untill it recover.

2.9.2 Parallel Database System

Parallel database systems architecture consists of a multiple central processing units (CPU) and data storage disk in parallel as shown in the figure. Hence, in such a system data processing speed is fast as well as input/output speed is also fast. The system, in which it needs to process an extremely large number of transactions per second, in such a system parallel database system is used.

Advantages of such sytem are given below :

- Useful in the applications, which have to process an extremely large number of transactions per seconds (of the order of thousands of transactions per seconds)
- Performance of such database system is very high.

The following figure shows a parallel database system.

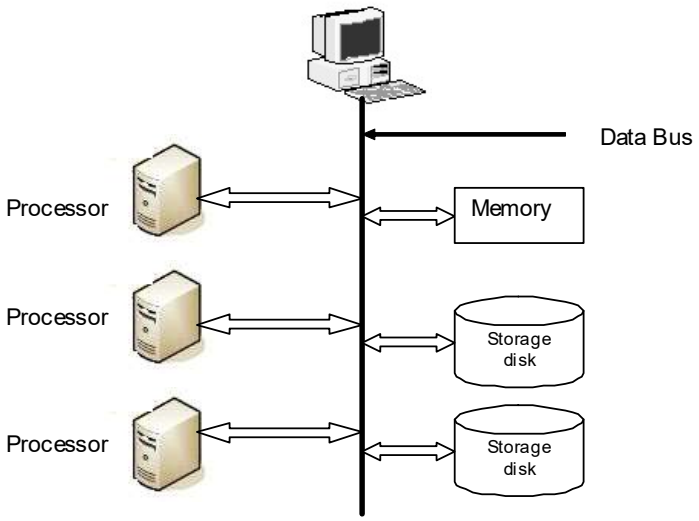


Fig.2.8: Parallel database system

2.9.3 Distributed DBMS

In a distributed database system, data are spread across a variety of different databases. These are managed by a variety of different DBMS

softwares running on a variety of different computing machines having different operating systems. These machines are actlually located on different sites and connected with some kind of communication networks as shown in the figure below. Thus, in a distributed database system, the organizations data might be distributed on different computers in such a way that data for one portion (or department) of the organization is stored in one computer and data for another department is stored in another computer. Each machine can have own data and applications and the users of one machine can access the data of several other computers.

The following figure shows a distributed database system.

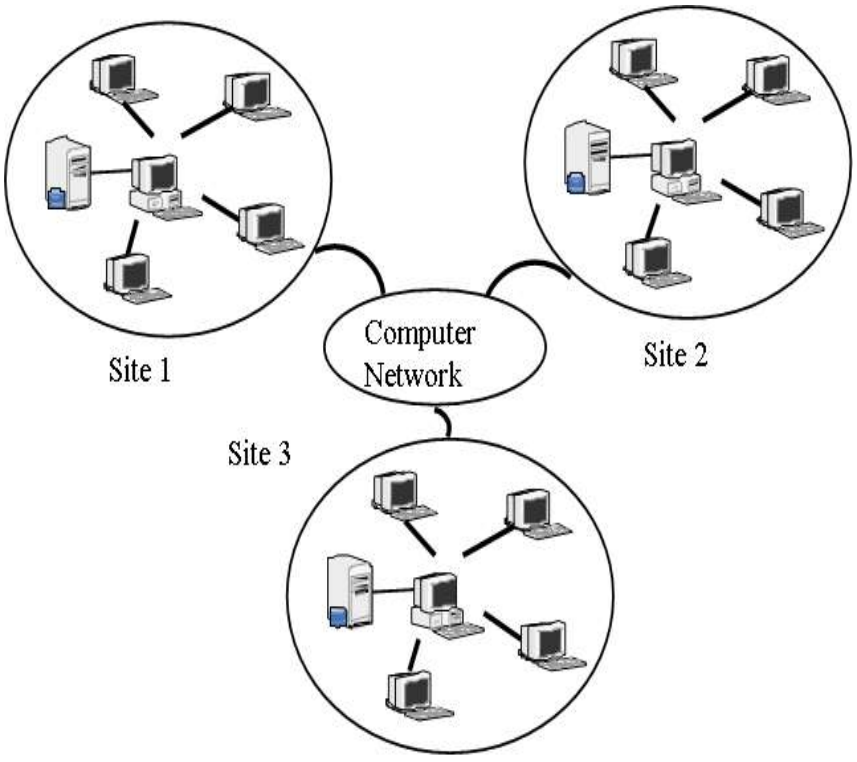


Fig.2.9: Distributed database

Advantages of such sytem are given below :

- efficiency and performancce of this system is high.
- a single database can be shared across several distinct client systems.
- As data volumes and transaction rates increase, the users can grow the system incrementally.

2.9.4 Client-Server DBMS

The client-server database system has two logical components namely - *client* and *server*. Clients are generally personal computers or workstations and the servers are the large workstations or mainframe computer system. The applications and tools of DBMS run on one or more client plateforms, while the DBMS softwares reside on the server. The server computer is called *backend* and the client computer is called *front-end*. The server and the clients are connected through networks. The clients send request to the server for performing some special tasks. The DBMS in the server side, in turn, process these requests and returns the results to the clients. The server handles parts of the job that are common to many clients, for example, database access and updates.

The following figure shows a client-server database model.

Advantages of such system are given below :

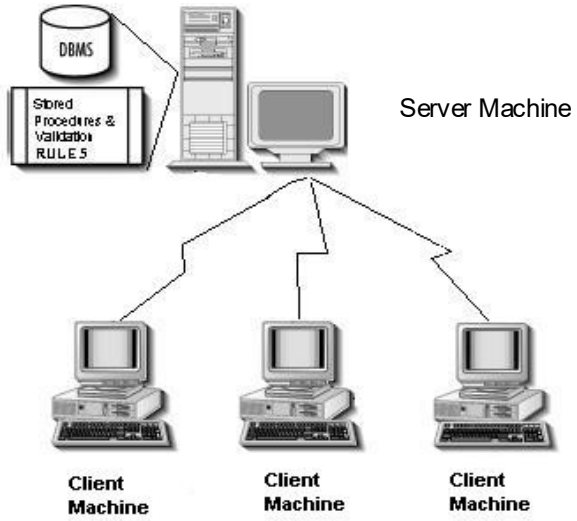


Fig 2.10 : Client-server database model

The advantages of such system are given below **:**

- Performance is high.
- A single database can be shared accross several distinct client system.

- More flexible as compared to the centralised system.
- Facilitates in more productive work by the users and making better use of existing data.

2.10 DATABASE ADMINISTRATOR

A database administrator (DBA) is a person or a group of person who is responsible for the environmental aspects of a database. A DBA is the central controller of the database system who designs database, controls and manages all the resources of database as well as provides necessary technical support for implementing policy decisions of database.

The role of a database administrator has changed according to the technology of database management systems (DBMSs) as well as the needs of the owners of the databases.

Some of the roles of the DBA may include

- Installation of new software — It is primarily the job of the DBA to install new versions of DBMS software, application software, and other software related to DBMS administration.
- Configuration of hardware and software with the system administrator — In many cases the system software can only be accessed by the system administrator. In this case, the DBA must work closely with the system administrator to perform software installations, and to configure hardware and software so that it functions optimally with the DBMS.
- Security administration — One of the main duties of the DBA is to monitor and administer DBMS security. This involves adding and removing users, administering quotas, auditing, and checking for security problems.
- Data analysis — The DBA will frequently be called on to analyze the data stored in the database and to make recommendations relating to performance and efficiency of that data storage.
- Database design (preliminary) — The DBA is often involved at the

preliminary database-design stages. Through the involvement of the DBA, many problems that might occur can be eliminated. The DBA knows the DBMS and system, can point out potential problems, and can help the development team with special performance considerations.

- Data modeling and optimization — By modeling the data, it is possible to optimize the system layouts to take the best advantage of the I/O subsystem.
- Responsible for the administration of existing enterprise databases and the analysis, design, and creation of new databases.



CHECK YOUR PPROGRESS

2. Select TRUE or FALSE in the following statements:
   (i) The conceptual view is a view of the totaldatabase content.
   (ii) User's view is also called external view.
   (iii) The database schema and an instance of the database are the same thing.
   (iv) A view of a database that appears to an application program is known as schema.
   (v) Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemes.
   (vi) A database is a computer-based record keeping system whose over all purpose is to record and maintain information.
3. Multiple Choice
   (a) A view of database that appear to an application program is known as –
      (i) schema (ii) subschema
      (iii) virtual table (iv) none of these
   (b) User's view is also called
      (i) external view (ii) conceptual view

(iii) internal view (iv) none of these

(c) Which of the following schemas defines the stored data structures in terms of the database model used -

(i) external (ii) conceptual

(iii) internal (iv) none of these

(d) Data is processed by using

(i) DDL (ii) DML

(iii) DCL (iv) DPL

(e) Immunity of the conceptual (or external) schemas to changes the internal schemas is referred to as

(i) physical data independance

(ii) logical data independence

(iii) both (i) and (ii)

(iv) none of these



2.11 LET US SUM UP

1. The traditional file approach to information processing has for each application a separate master file and its own set of application programs, COBOL language used to write these application programs.
2. A database is a single organized collection of instructed data, stored with a minimum of duplication of data items so as to providea consistent and controlled pool of data.
3. A database management system (DBMS) is a collection of programs that enables the users to store, modify and extract information from a database as per the requirements. DBMS is an intermediate layer between programs and the data. Programs access the DBMS, which then accesses the data.
4. According to the ANSI/SPARC architecture of a database system the whole database is divided into the following three levels :
   - External level or view level
   - Conceptual level
   - Internal level or physical level
5. Logical data independence indicates that the conceptual schema can

be changed without affecting the existing external schema.

6. Physical data independence indicates that the physical storage structures or devices used for storing the data could be changed without necessitating a change in the conceptual view or any of the external views.
7. DBMS provide appropriate languages and interfaces for each category of users to express database queries and updates.
8. Following languages are used for specifying database schemas :
   - Data definition language (DDL)
   - Storage definition language (SDL)
   - View definition language (VDL)
   - Data manipulation language (DML)
9. DBMS can be classified according to the number of users, the database site locations and the type and extent of use.

   On the basis of site locations, the followings are types of DBMS :
   - Centralised DBMS
   - Parallel DBMS
   - Distributed DBMS
   - Client/Server DBMS

   On the basis of the type and extent of use DBMS are of following types :
   - Transactional DBMS
   - Decision support DBMS
   - Data Warehouse
10. DBA is database administrator who is responsible for maintaining database. A DBA provides the necessary technical support for implementing policy decisions of database.



2.12 FURTHER READING

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of database systems*. Pearson.
2. Date, C. J. (2006). *An introduction to database systems*. Pearson Education India.



2.13 ANSWER TO CHECK YOUR PROGRESS

1. a. (iv) b. (iv) c. (ii) d. (i) e. (i)
2. (i) False (ii) True (iii) False (iv) False (v) True (vi) True
3. a. (ii) b. (i) c. (ii) d. (ii) e. (i)



2.14 MODEL QUESTIONS

1. What is the file based approach of database? Explain its limitations?
2. Explain the three levels of database architecture. What are its objectives?
3. Define data independence and its types? How is data independence achieved?
4. State the advantages of DBMS?
5. Discuss the main disadvantages of a Traditional file approach?
6. Discuss the main disadvantages of DBMS?
7. Differenciate between DBMS approach and the traditional file approach.
8. Mention the differences between text files and database files. Why are database files preferred in a commercial organization?
9. Write short notes on :
   (i) Data independence
   (ii) Database
   (iii) DBMS
   (iv) DBMS Architecture
   (v) Client-server database model
   (vii) Distributed database system
   (vii) Physical Data Independence
   (viii) traditional File Approach
   (ix) Centralised database system
   (x) DBMS language
10. What is logical data independence and why is it important?
11. Explain the difference between logical and physical data independence.
12. Describe the three levels of data abstraction?
13. What do you mean by Database Language? What are the different types of data base language?
14. Explain the role of a Database Administrator.

***

UNIT 3: DATA MODEL

UNIT STRUCTURE

3.1 Learning Objectives
3.2 Introduction
3.3 Data Models
  3.3.1 Conceptual Model
  3.3.2 Logical Model
  3.3.3 Physical Model
3.4 Entity Relationship Model
  3.4.1 Entity, Attribute and Relationship type
  3.4.2 Degree of Relationship type
  3.4.3 Constraints on Relationship type
  3.4.4 Weak and Strong Entities
3.5 Conversion of ER model into Relational Schema
3.6 ER Modelling Symbols
3.7 Let Us Sum Up
3.8 Answers to Check Your Progress
3.9 Further Reading
3.10 Model Questions

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe different data models
- identify the categories of data model used in database management system
- identify the symbols to describe the ER model
- design the ER diagram with its constraints
- describe conversion procedure from ER diagram to relational databases and vice versa

3.2 INTRODUCTION

In our previous units we have discussed the various approaches of database system and their architecture. We have come across the schema diagram of DBMS architecture, their types and also their merits and demerits.

Now in this unit, we will be able to learn a brief description of all the three approaches and the various types of data model that are categorised on the basis of these concepts. We will also discuss in detail the design approach of Entity Relationship (ER) model with all their constraints. ER model is primarily a semantic model and is very useful in creating raw database design that can be further normalised. This is usually used for real world application development process. Along with this we will discuss the basic conversion procedure of ER model into relational schema with suitable example. We will conclude this unit with the modelling symbol used in ER diagram.

3.3 DATA MODELS

There are many basic structures that exist in a database system to organise the data. One of the fundamental characteristics of the database approach is that it provides some level of data abstraction by hiding details of data storage that are not needed by most database users. Data model is the main tool to provide this abstraction. A data model is a set of concept that can be used to describe the structure of a database. Each structure of a database has its own data type, relationships of one data structure to another and constraints that should hold on the data. Most data models also include a set of operation for specifying retrievals and updates on the database. So, a data model defines

- The logical data structure
- Data relationships
- Data consistency constraints.

Data models are categorised in different ways. The most general category is on the basis of the concepts they provide to describe the database structure. They are: *Conceptual Model, Logical Model, and Physical Model.*

3.3.1 Conceptual Model

Conceptual model is also called **High Level Data Model**. It provides the concepts that are close to the way by which many users perceive data. It uses concepts such as entities, attributes and relationship types. An *entity* is an object which is used to represent a database. An *attribute* is a property that describes some aspects of an object. *Relationship* is the relation between entities that is easily represented in high level data models. It is sometimes called **Object-Based Model** because it uses objects as key data representation components that contain both data members/values and operations that are allowed on the data. The interrelationships and constraints are implemented through objects, links and message passing mechanisms. Object-Models are useful for databases where data interrelationships are complex, for example, Computer Assisted Design based components. The popular high level data model is Entity-Relationship Model.

3.3.2 Logical Model

Logical model is also called Implementation data model. It provides concepts understandable by the end users but is not too far removed from the way the data is organised within the computer. It hides some details of data storage but can be implemented on a computer system in a direct way. This model is sometimes called **Record-based data model** because it uses records as the key data representation components. It is used most frequently in current commercial DBMSs and includes the three most widely used data models - *Relational*, *Network* and *Hierarchical data model*.

3.3.3 Physical Model

It is the low-level data model. It provides concepts that describe the details of how data is stored in the computer by representing information such as record formats, record orderings and access paths. An access

path is a structure that makes the search for particular database records much faster.



CHECK YOUR PROGRESS

1. State true or false (T/F) of the following:

a) Data model gives logical as well as physical data structure.

b) Conceptual data model is also called object-based model.

c) High level data model uses records as the key data representation component.

d) Low level data model describes the details of how to store data in computer.

e) Logical data model provide concept which is complicated for end users.

f) Implementation data model is also a record-based data model.

g) Access path can search database record much faster.

h) Three most widely used data models belongs to the physical data model

3.4 ENTITY RELATIONSHIP MODEL

Entity-Relationship (ER) model is a high-level conceptual data model developed to facilitate database design. The ER model concepts are designed to be closer to users' perception of data and are not meant to describe the way in which data will be stored in the computer. It is the generalised data model. Although ER model has some means of describing the physical database model, it is basically useful in the design of logical database model. This analysis is then used to organise data as a relation, normalizing relation and finally obtaining a relational database model.

Some of the main features of ER model are:

- It is a high level conceptual data model.
- It allows describing the data involved in a real-world enterprise in

terms of objects and their relationships.

- This is widely used to develop an initial design of a database.
- Describes data as a collection of entities, relationships and attributes.

3.4.1 Entity, Attribute and Relationship type

ER model uses three features to describe data. These are: *entity*, *attribute* and *relationship*.

Entity:

- An entity is an object which specifies distinct things in the real world items, e.g., car, book, table, student, employee etc.
- It need not be a physical entity; it can also represent a concept in real world like project, loan, account etc.
- It represents class of things, not any one instance. For example, 'STUDENT', 'EMPLOYEE' etc.

Each entity has attributes. For example, an employee may be described by the employee's name, employee id, age, address, designation and salary. In figure 3.1(a) and 3.1(b), two entity types EMPLOYEE and STUDENT, are shown with their different attributes.

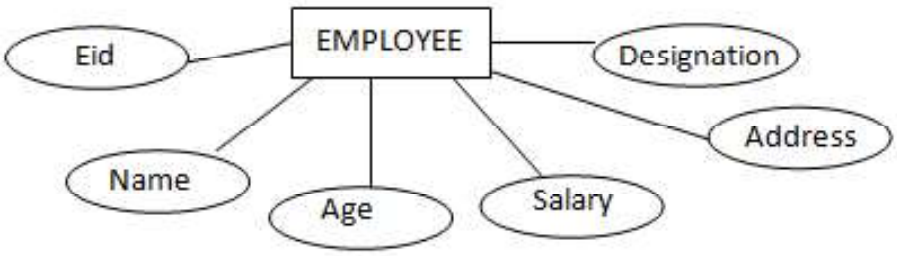


Figure 3.1(a): EMPLOYEE entity type with six different attributes.

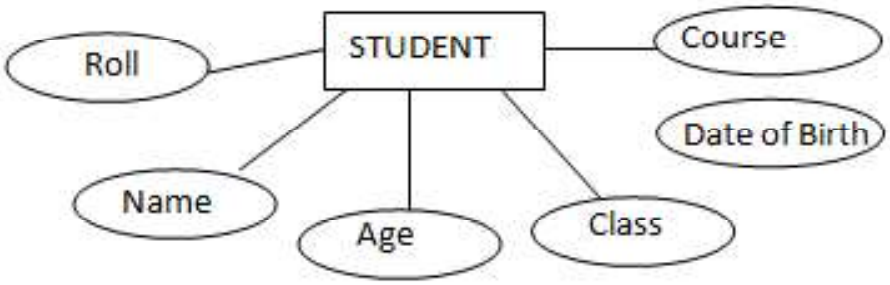


Figure 3.1(b): STUDENT entity type with six different attributes.

Entity Type:

A collection of a similar kind of entities having same attribute is called an ***Entity Set*** or ***Entity Type***. For most databases there may be many entity types. An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name. Attribute names are enclosed in ovals and are attached to their entity type by straight lines. [Fig. 3.1 (a) and Fig. 3.1 (b)]

Key Attributes of an Entity Type: An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely. For example, Roll attribute is a key of the STUDENT entity type.

Attributes:

The properties of an entity that describes the specific feature of the entity is called *attribute*. In case of a student entity, 'Roll No', 'Student's name', 'Age', 'Marks' etc. are attributes.

Each entity posseses values for their attributes. For example, in case of STUDENT entity type, the attributes are Roll, Name, Age, Class, Date of Birth and Course. The values for those attributes may be like :

Roll =1, Name = Rishi , Age =17, Class = Twelve, Date of Birth = 21/07/2002 , Course= Science.

The attribute values are different for differnet students. There may be some common attribute values for two different students.

Value Sets (Domains) of attribute:

Each simple attribute of an entity type contains a possible set of values that can be attached to it. This is called the domain of an attribute. An attribute cannot contain a value outside this domain.

For example, in the Figure 3.1 (a), if the range of attribute 'age' allowed is in between 20 and 50, we have to specify the value set of the age of EMPLOYEE to be the set of integer numbers from 20 to 50. Similarly, we can set the value set for 'Name' attribute as the set of strings of alphabetic characters separated by blank characters and so on.

Types of attributes:

Attributes attached to an entity can be of various types.

i) **Simple (atomic) attribute:** The attribute that cannot be further divided into smaller parts and represents the basic meaning is called the simple attribute. For example, pin code of a person.

ii) **Composite attribute:** Attribute that can be further divided into smaller unit and each individual unit contains a specific meaning. It is composed of several more basic attributes. For example, address of a person is composed of zip code, street, district etc.

iii) **Single valued attribute:** Attribute having a single value for a particular entity. For example, a person entity having one value for age. So age is a single-valued attribute.

iv) **Multi valued attribute:** Attribute that has a set of values for the same entity is called a multi valued attribute. Different entities may have different number of values for this kind of attribute. For multi valued we much also specify the minimum and maximum number of values that can be attached. For example, college degree attribute for a person.

v) **Stored attribute:** Attributes that are directly stored in the database. For example, birth date attribute of a person.

vi) **Derived attribute:** attributes that are not stored directly but can be derived with the help of some other attribute. For example, a person's age is derived from date of birth of that person.

Relationship:

In most databases there will be more entities and entities are joined with each other by some relations called relationships between entities. For example, entity EMPLOYEE has the relationship with entity DEPARTMENT. A relationship can be defined as:

- A connection or set of associations, or
- A rule for communication among entities.

Relationship sets:

A relationship set is a set of relationships of the same type. Collection of all the instances of relationship forms a relationship set called *relationship type.*

For example, consider a relationship type WORKS_FOR between the two entities EMPLOYEE and DEPARTMENT, which associates each employee with the department the employee works for. Each relationship instance in WORKS_FOR associates one employee entity and one department entity.

3.4.2 Degree of Relationship Type

The degree of a relationship type is the number of participating entity type. A relationship type of degree two is called binary relation and of degree three is called ternary relation. An example of binary and ternary relationship type are explained in Figure 3.2, where the degree of WORKS_FOR relationship type is two and for SUPPLY the degree is three.

(a)

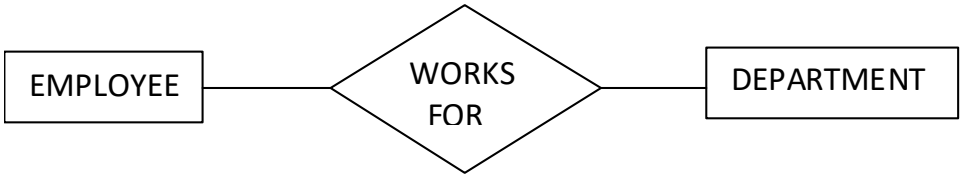


Figure 3.2: (a) binary relationship WORKS_FOR

(b)

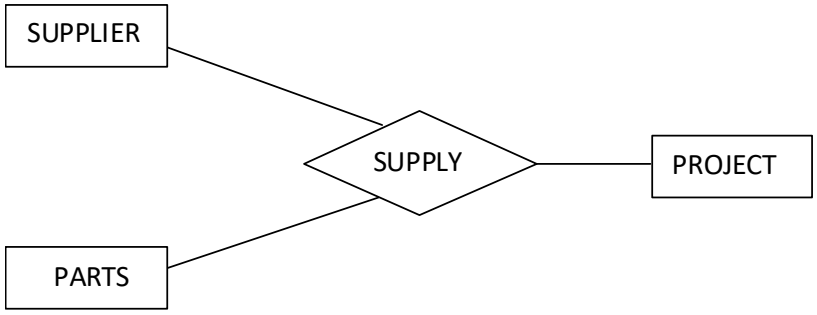


Figure 3.2 (b) ternary relationship SUPPLY

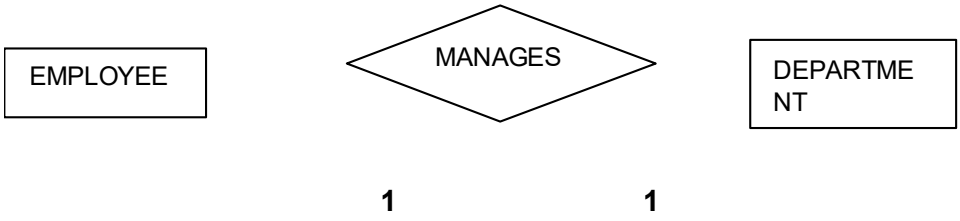
3.4.3 Constraints on Relationship type

Relationship types usually have certain constraints that limit the possible combinations of entities participating in relationship instances. These constraints are determined from the mini world situation that the relationships represent. The two main types of relationship constraints that occur relatively frequently are cardinality ratio and participation constraints.

a) Cardinality Ratio:

This constraint specifies the number of relationship instances that an entity can participate in. Common cardinality ratios for binary relationship types are **1:1, 1:N** and **M:N**.

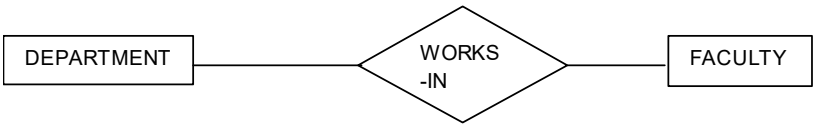
- **One-to-one (1:1):** This is the relationship where an entity in an entity type 'A' is associated with at most one entity of the entity type 'B' and an entity in entity type 'B' is associated with at most one entity in entity type 'A'. For example, the relationship between employee and department.



It is 1:1 relationship type, because it is assumed that at least one employee should be there to manage one department and one department should have one manager.

Similarly, we can define the relationship between college and principal because we know that one college can have at most one principal and one principal can be assigned to only one college.

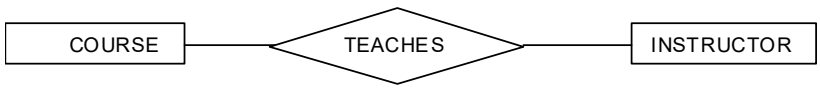
- **One-to-many (1:n):** If an entity in entity type 'A' is associated with any number of entities in entity type 'B' and an entity in entity type 'B' is associated with at the most one entity in entity type 'A', then this type of ratio is called 1:n cardinality ratio. For example, the relationship between department and faculty.



1 **N**

Here, in the example, it is considered that a particular department may have more than one faculty members but a faculty member can be an employee of only one department.

- **Many-to-one (m:1):** In this type of ratio, an entity in entity type 'A' is associated with at most one entity in entity type 'B' or an entity in entity type 'B' is associated with any number of entity in entity type 'A'. For example, the relationship between course and instructor.

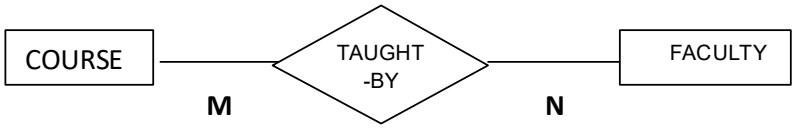


M **1**

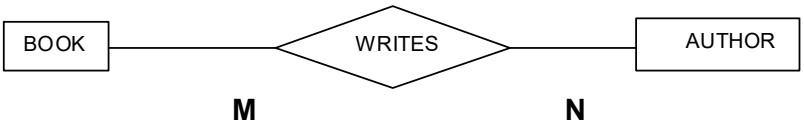
It is assumed that an instructor can teach various courses but a course can be taught only by one instructor.

- **Many-to-many (m:n):** Entities in entity type 'A' and 'B' are associated with any number of entities from each other. For example, if we assume that one faculty member can be assigned to teach many courses and one course may be taught by many faculty

members, then the relationship taught-by will be a many-to-many relationship.



Like relationship between book and author, if we assume that one author can write many books and one book can be written by more than one authors, then it will be many-to-many relationship.



b) Participation Constraints:

The participation constraints specify whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints: **total** and **partial**.

Total participation constraints: When all the entities from an entity set participate in a relationship type, is called **total participation**. It is denoted by double line ' ═══ ' sign.

Partial participation constraints: When it is not necessary for all the entities from an entity set to participate in a relationship type, it is called partial participation and is denoted by '___' sign.

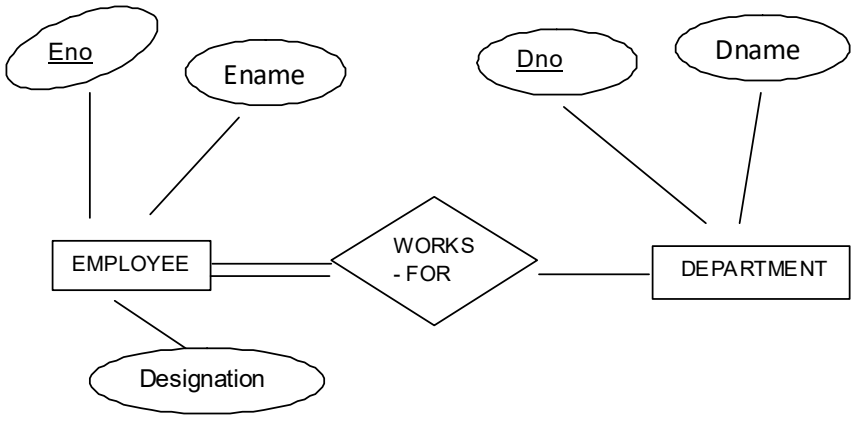


Figure 3.3: Total and Partial participation constraints

In the Figure.3.3, the relationship WORKS-FOR has two entities EMPLOYEE and DEPARTMENT. The participation constraints of entity EMPLOYEE is total and of entity DEPARTMENT is partial with relationship type WORKS-FOR, if we assume that,

i) All employees works for any one or more department, then if it is a total participation constraint.

ii) There are some departments which do not have any employee. It means that all departments of DEPARTMENT entity are not dependent on the entity EMPLOYEE via relationship WORKS-FOR. So, it is a partial participation constraint.

3.4.4 Weak and Strong Entities

Entities are of two types: *strong entities* and *weak entities*.

i) Strong entity type: The entity type containing a key attribute is called a strong entity type or a regular entity type. For example, if the entity 'student' has a key attribute 'Roll No' which is used to uniquely identify the entity 'student', then the entity 'student' is called a strong entity.

ii) Weak entity type: The entity type without having any key attribute is called weak entity type. For example, an employee wants to keep track of the information of their dependents and the attributes of dependent entity is name, sex, and relationship with employee. Here, the entity dependent does not have any key attribute; hence this type of entity is called weak entity.



CHECK YOUR PROGRESS

2. State true or false (T/F) for the following

a) An entity is an object used to represent things.

b) Entities having key attributes are called weak entity.

c) Strong entity is also called regular entity.

d) Domain may be the range of values used against attribute.

e) Composite attribute cannot be divided into smaller parts.

3. Fill in the blanks in of the following studences with appropriate word given below:

attributes, many-to-many, diamond, association, ER.

a) An _____ of several entities in an Entity-Relation model is called relationship.

b) The various kinds of data that describes an entity are known as its _____.

c) A weak entity set is represented by a doubly outlined rectangle in the _____ diagram.

d) In an ER diagram a _____ represents a relationship

e) A _____ relationship describes entities that may have many relationships in both the directions



ACTIVITY -1

1. Consider a COLLEGE database that keeps track of students, faculty, department and courses organised by various departments. College contains departments and each department is assigned a unique id and name. Some faculty members are also appointed to each department and one of them works as head of the department.

a) Find out the possible entities and their relationship types.

b) List out the possible key attributes of each entities.

c) Give the cardinality ratio between faculty and department entities.

d) Show the participation constraints of entities students and courses needed by the departments

3.5 CONVERSION OF ER MODEL INTO RELATIONAL SCHEMA

For every ER diagram we can construct a relational database which is a collection of tables called schema diagram. Following are the set of

steps to convert ER diagram to a relational schema.

Conversion of entity sets:

a) For each strong entity type E in the ER diagram, we create a relation R containing all the simple attributes of E. The key attribute of the relation R will be one of the key attribute of R. For example, the STUDENT entity having Roll No as a key attribute, the relational schema will be as shown below

Roll No : primary key	**name**	**address**

Similarly, we can convert other entities also into corresponding relational schema.

b) For each weak entity type W in the ER diagram, we create another relation R that contains all the simple attributes of W. If E is an owner entity of W then the key attribute of E is also included in R. This key attribute of R is a set as a foreign key attribute of R. Now the combination of primary key attribute of owner entity type and partial key of the weak entity type will form the key of the weak entity type. For example, there is a weak entity GUARDIAN, where any key field of student entity RollNo has been added.

RollNo	Name	Address	Relationship
(Primary Key)			

Conversion of relationship sets:

Binary relationships:

a) One-to-one relationship: For each 1:1 relationship type R in the ER diagram involving two entities E1 And E2 we choose one of the entities (say E1) preferably with total participation and add primary key attribute of another entity E2 as a foreign key attribute in the table of entity (E1). We will also include all the simple

attributes of relationship type R in E1 if any. For example, considering DEPARTMENT entity having attributes DNO, DNAME and one Head_of 1:1 relationship between FACULTY and DEPARTMENT. We choose DEPARTMENT entity having total participation and add primary key attribute ID of FACULTY entity as a foreign key in DEPARTMENT entity named as Head_ID. Then the relational schema will be as:

DEPARTMENT

Dno	Dname	Head_Id	Date_from

b) **One-to-many relationship:** For each 1:N relationship type R involving two entities E1 and E2, we identify the entity type (say E1)at the n-side of the relationship type R and include primary key of the entity on the other side of the relation (say E2) as a foreign key attribute in the table of E1. We include all simple attributes or simple components of a composite attributes of R (if any) in the table of E1.

For example, consider a relationship WORKS_ON between the DEPARTMENT and FACULTY entities. For this relationship choose the entity at N side, i.e., FACULTY and add primary key attribute of another entity DEPARTMENT, i.e., DNO as a foreign key attribute in FACULTY. The relational schema for this relationship will be as:

FACULTY (CONTAINS WORKS_ON RELATIONSHIP)

Id	Name Address	Basic_pay	DNO

c) **Many-to-many relationship:** For each M:N relationship type R, we create a new table (say S) to represent R. We also include the primary key attributes of both the participating entity types as a foreign key attribute in S. Any simple attributes of the M:N rela-

tionship type or simple components of a composite attribute is also included as attributes of S. For example, consider a M:N relationship TAUGHT_BY between entities COURSE and FACULTY should be represented as a new table. The structure of the table will include primary key of COURSE and primary key of FACULTY entities. The relational schema for TAUGHT_BY relationship will be a new table as:

TAUGHT_BY

ID (Primary key of FACULTY table)	Course_ID (Primary key of COURSE table)

n_ary Relationship:

For each n_ary relationship type R where n>2, we create a new table S to represent R. We include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the n_ary relationship type (or simple components of complete attributes) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

Multi valued attributes:

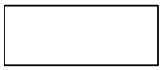
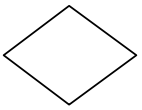
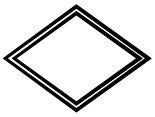
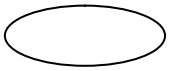
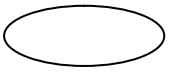
For each multi valued attribute 'A', we create a new relation R that includes an attribute corresponding to the primary key attribute k of the relation that represents the entity type or relationship type that has as an attribute. The primary key of R is then combination of A and k.

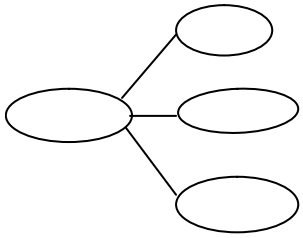
For example, if the STUDENT entity has attributes Roll No, Name and Phone No with Roll No as a primary key and Phone No as a multi valued attribute, then we will create a table PHONE(Roll No, Phone No) where primary key is the combination. In this

case the STUDENT table need not have the attribute Phone No; instead it can be simply Roll No and Name.

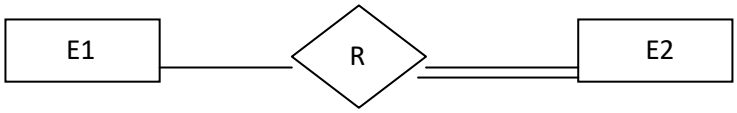
3.6 ER Modelling Symbols

We can express the overall logical structure of a database graphically by using ER diagram. ER diagram is composed according to the following modelling symbols:

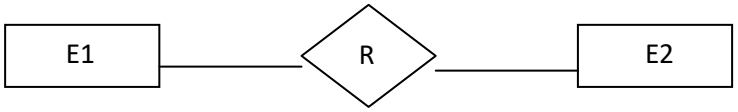
Symbol	Meaning	Symbol	Meaning
	Entity		Relationship
	Weak Entity		Identifying Relationship
	Attribute		Multi Valued Attribute
	Key Attribute		Derived Attribute



Composite Attribute



Total participation of E2 in R



1 **N**

Cardinality ratio 1:N for E1:E2



CHECK YOUR PROGRESS

4. State whether the following are true or false (T/F)

a) For weak entity, only the key attributes of owner entity will form relational table.

b) For each and every ER diagram, we cannot construct a relational schema.

c) For strong entity type, all the keys of those entities will form the relational schema.

d) We cannot form a relational schema for relationships of entities.

e) For 1:1 relationship, only the key attributes of entities of both sides will form relational schema.

5. Fill in the blanks:

a) We include all simple attributes in relational schema for _____ relationship type.

b) For each _____ in the ER diagram, we create another relation that contains all simple attributes.

c) For _____ relation, the primary keys only form a new relation.

d) Relational database is a collection of tables called



ACTIVITY 2

Consider the ER diagram of COLLEGE given in the Activity 1, and construct the corresponding relational schema for that.



3.7 LET US SUM UP

The ER model explained in this unit covers the basic aspects of ER modelling, database designing procedure to store data in a database with their basic symbols to represent.

- A data model is a set of concept that can be used to describe the structure of a database.
- Data models are of three types: Logical, Physical and Conceptual.
- The most often used data models in current commercial DBMSs are Relational, Network and Hierarchical models.
- E-R is a high-level conceptual data model. It is basically useful in the design of logical data base model. It describes data as a collection of entities, relationships and attributes.
- An entity is an object which specifies distinct things in the real world items, e.g., car, book, table etc. A collection of a similar kind of entities having same attributes is called an Entity Set or entity type.
- An entity having key attribute is called strong entity or a regular entity and without key attribute is called weak entity.
- The properties of entity that describes specific feature of entity is called attribute. These are of different types that described in this unit
- The relations by which entities are joined each other is called relationships between entities.
- Representation of entities with tabular form is called schema diagram. For every ER diagram we can construct a relational database which is a collection of tables. The attributes of an entity will be also attributes of the schema diagram.
- For each relationship type there will be one schema diagram.



3.8 ANSWERS TO CHECK YOUR PROGRESS

1. a) False b) True c) False d) True
   e) False f) True g) True h) False

2. a) True b) False c) True d) True
   e) False

3. a) association b) attributes c) E-R d)diamond
   e) many-to-many

4. a) True b) False c) True d) False
   e) False

5. a) 1:N b) Weak entity type c) M:N
   d) schema diagram



3.9 FURTHER READING

1. Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems. Pearson.
2. Date, C. J. (2006). An introduction to database systems. Pearson Education India.



3.10 MODEL QUESTIONS

Q1. What do you mean by Data Model? What does it define?
Q2. Describe conceptual, logical and physical data model.
Q3. Design an E-R diagram for airline reservation system consisting of flights, aircrafts, airports, fares, reservations, tickets, pilot, crew and passengers. Clearly highlight the entities, the relationships, the primary keys and the mapping constraints.

4. What do you mean by Constraints of Relationship type? Explain with example the constraint of different cardinality ratio.
5. Discuss about participation constraints used in E-R diagram design. Distinguish between Total participation constraints and Partial participation constraints.
6. What is Cardinality Ratio in ER diagram?
7. Discuss the procedure of converting E-R model into relational schema.

9 Can you convert a relationship type of an E-R model into corresponding relational schema? If Yes, how?

10. Explain E-R modelling symbols.

****

UNIT 4: RELATIONAL MODEL

UNIT STRUCTURE

4.1 Learning Objectives
4.2 Introduction
4.3 Relational Data Model Concept
    4.3.1 Relational Schema and Instances
4.4 Integrity Constraints
    4.4.1 Entity Integrity Constraints
    4.4.2 Referential Integrity Constraints
    4.4.3 Domain Constraints
4.5 Let Us Sum Up
4.6 Answers to Check Your Progress
4.7 Further Reading
4.8 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- define Relational model
- define the terminologies used in Relational model
- describe the Schema design of a database
- define various Integrity Constraints
- describe how data are organised in the form of tables

4.2 INTRODUCTION

In the previous unit, we have discussed the properties of data models and the designing procedure of Entity-Relationship model with symbols. We also discussed the conversion procedure of ER model to its Relational data model.

This unit focuses on Relational model. Most of the commercial DBMS products available in the industry are relational at core. In this unit we will discuss the terminology and operations used in relational model. The restrictions associated with the formulation of relation table will also will be discussed in this unit.

4.3 RELATIONAL DATA MODEL CONCEPT

We have already come to know across that a data model in a database system is basically a structure or an organisation of data and a set of operations on that data. Relational data model is one of the traditional data model. This is the most widely used database model that represents data as well as relationship among the data in the form of tables. This is a very simple model and is based on a proven mathematical theory. In this model, a database is represented as a collection of "Relations", where each relation is represented by a two dimensional table. This model is most commonly used in real world due to its simplicity. Following figure (4.1) represents a simple "STUDENT" relation:

R_NO	S_NAME	ADDRESS	MARKS
10	Sanjib Kaur	Block -4, Noonmati	69
12	Padip Sen	Ganeshguri	75
15	Bipul Prasad	Bamunimaidan	58

Fig. 4.1: A sample STUDENT relation

Properties of a table

A table should contain the following properties:

a) Each entry in a table represents one data item and two column headings with the same name are not allowed.
b) In each column, the data items are of the same data type.
c) Each column is assigned with a distinct heading.
d) All rows are distinct; duplicate rows are not allowed.

e) Both the rows and the columns can be viewed in any sequence at any time without affecting the information.

A relational database model uses a collection of tables to represent both the data and the relationships among those data items. Each table has multiple columns and each column has a unique name.

Properties of RDBMS

A relational Database Management System (RDBMS) has the following properties:

a) Stores data in the form of tables.
b) Does not require the user to understand its physical implementation.
c) Provides information about its contents and structure in the system table.
d) Supports the concept of NULL values.

Advantages of Relational Data Model:

Following are some of the **advantages** of relational model:

a) Ease to use: Convenient to define and query the database as tables contain rows and columns. It is quite natural even for first time users.
b) Flexibility: Data can manipulate easily with the help of relational operations.
c) Precise: Since it uses mathematical operations, it can ensure accuracy and less ambiguity as compared to other models.
d) Security: security control and authorization can also be implemented more easily. It has the user's own authorization controls.
e) Data Independence: Data independence is achieved more easily with normalization structure used in a relational database.
f) Data Manipulation Language: The possibility of responding to query on relational algebra and relational calculus is easy in the relational database approach.

The basic terminologies used in relational database model are:

- **Tuple (record) and attribute:**

Each row in a table is called a tuple and a column name is called an attribute. For example, Figure 4.1 represents a STUDENT relation where ROLL NO, NAME, ADDRESS and MARKS are attributes and each entry against these attributes is called tuple of relation STUDENT.

- **Domain:**

A domain is a collection of all possible values from which the values for a given column or attribute is drawn. So, every attribute in a table has a specific domain. Values to these attributes cannot be assigned outside their domains. For example, the domain of attribute NAME is the set of all alphabetic string of finite length and the domain of a MARKS attribute should not be greater than 100 for the relation STUDENT in Figure 4.1.

- **Relation:**

The table with all tuples and attributes is called relation. It has three components: **Name** that is represented by the title of the relation, **Degree**, the number of column associated with the table and the **Cardinality**, the number of rows in the table. For example, figure 4.1 represents a relation named STUDENT of degree 4, because it has total four attributes, and the cardinality for this relation is 3(number of rows).

4.3.2 Relational Schema and Instances

A relation consists of Relational Schema and Relation Instances.

Relational Schema: A schema specifies the relation name, its attributes and the domain of each attribute. The description of a database is called database schema. It is specified during database design and is not expected to change frequently. If R is the name of a relation and A1, A2, A3..... An are the attributes of R, then R (A1, A2,.....An) is called a relational schema. Each attribute takes values from a specific domain D. A displayed schema is called a schema diagram. It shows only the specified

data items, not the data types or relationships among various other files. For example, the relational schema diagram for the relation "STUDENT" is given below:

STUDENT

R_NO	S_NAME	ADDRESS	MARKS

where, STUDENT is the name of the Relation and R_NO, S_NAME, ADDRESS, and MARKS are four different attributes that represent the relational schema.

Relation Instance: The data in the database at a particular moment in time is called a database instance. It is also called an occurrence or state. Many database instances are corresponding to a particular database schema.

We can define the relational instances as,

A relation instance denoted as 'r' is a collection of tuples for a given relational schema. The relation state of the relational schema R (A1, A2,.....An) is denoted as r(R) is a set of n-tuples.

The relation schema is also called ***'intension'*** and relation state is called ***'extension'.***



CHECK YOUR PROGRESS

1. Fill in the blanks for the following.

i. In a relational database model, the columns of a table are called _____.

ii. _____ is the rows in a table.

iii. _____ is a collection of related files.

iv. The number of columns in a table is the _____ of the relation.

v. _____ is the number of rows in a table.

vi. An entity is an object that is distinguishable from other objects by

a specific set of _____

vii. Collection of tuples for a relational schema at a particular time is called _____

viii. A relation state of a relational schema R denotes as _____ if r is the collection of tuples.

ix. A relation state is also called _____

x. Relational schema contains name and _____of that relation.

4.4 INTEGRITY CONSTRAINTS

The term ***integrity*** refers to the accuracy or correctness of data in the database schema and is expected to hold on every database instance of that schema. Relational model includes two general integrity constraints. They are: Entity Integrety Constraints (Rule 1) and Referential Integrity Constraints (Rule 2)

4.4.1 Entity Integrity Constraints (Rule 1)

Entity Integrity Constraints state that no primary key value can be NULL. This is because we use the primary key value to identify individual tuples in a relation. It ensures that instances of the entities are distinguishable i.e., they must have a unique identification of some kind. Primary keys perform that unique identification function in a relational database.

4.4.2 Referential Integrity Constraints (Rule 2)

Referential Integrity Constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations (not necessarily be distinct). It uses a concept of foreign key which will be explained more details in the subsequent unit. Informally, it states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. Considering the following relations,

EMPLOYEE

p.k **f.k**

ENO	ENAME	DNO
101	Robert	10
102	Smith	12
103	Robindra	12
104	John	10

DEPARMENT

p.k

DNO	DNAME	LOCATION
10	Comp. Sc.	Jalukbari
12	Electronic Sc.	Guwahati

Fig. 4.2 relational database table showing referential integrity

In figure 4.2, EMPLOYEE and DEPARTMENT are two relations where ENO and DNO are primary keys respectively. Here the attribute DNO of EMPLOYEE table is a foreign key that gives the department number for which each employee works. Hence its value in each EMPLOYEE tuple must match the DNO value of some tuple in the DEPARTMENT relation.
To specify the Referential Integrity constraints between two relations R1 and R2, the following properties have to be satisfied:

a) The Foreign key attribute of a relation have to be same domain with the primary key of the other relation.
b) The value of foreign key in a tuple t1 of relation R1 occur either a value of primary key of tuple t2 in R2 or to be NULL.

4.4.3 DOMAIN CONSTRAINTS

It specifies that each attribute in a relation must contain an atomic value only from the corresponding domains. The data types for commercial RDBMS domains are:

- Standard numeric data types for integer
- Real numbers
- Characters
- Fixed length and variable length strings

Thus, domain constraint specifies the condition that we want to put on each instance of the relation. So, the values that appear in each column must be drawn from the domain associated with that column.



CHECK YOUR PROGRESS

2. Fill in the blanks for the following

i. In a relational database, _____ are not allowed to have null values.

ii. In a relational database, a referential integrity constraint is specified with the help of _____.

iii. The _____ in a record is a unique data item.

iv. In a relation, each _____ name must have different meaning.

v. According to integrity rule 1, two _____ should be distinguishable from each other.

3. Write True or False.

i. Prime attribute have unique identifier.

ii. The key field may be NULL in relational database.

iii. Foreign key concept is used to explain Integrity Rule 1.

iv. Domain constraint specifies the values related to

a. instances.

v. According to referential integrity constraints, values of foreign key field of that table are dependent on the values of primary key field of another table



4.5 LET US SUM UP

- Relational model is a simple model in which database is represented as a collection of "Relations", where each relation is represented by a two dimensional table. Each table has multiple columns, and each column has a unique name.
- In Relational Database Model each row in a table is called a tuple and a column name is called an attribute. The table with all tuples and attributes is called relation.
- A domain is a collection of all possible values from which the values for a given column or attribute is drawn.
- A Relational Table has three components: Name that represent by the title of the relation, Degree, the number of column associated with the table and the Cardinality, the number of rows in the table.
- A Relational Schema of Relational Model specifies the relation name, its attributes and domain of each attribute and the Instances are collection of tuples for a given relational schema.
- The relation schema is also called 'intension' and relation state is called 'extension'.
- Integrity is the accuracy or correctness of data in the database schema and is expected to hold on every database instance of that schema. Relational model includes two general.
- Entity Integrity Constraints states that no primary key value can be NULL.
- Referential Integrity Constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations.
- Domain Constraint specifies that each attribute in a relation must contain an atomic value only from the corresponding domains.



4.6 FURTHER READING

1. Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems. Pearson.
2. Date, C. J. (2006). An introduction to database systems. Pearson Education India.
3. Singh, S. K. (2011). Database systems: Concepts, design and applications. Pearson Education India.



4.7 ANSWERS TO CHECK YOUR PROGRESS

1. i) Attribute ii) Tuple iii) Database iv) Degree
   v) Cardinality vi) Attribute vii) Relation instance
   viii) r(R) ix) Extension x) attribute

2. i) primary key ii) foreign key iii) key field
   iv) attribute v) entities

3. i) True ii) False ii) False
   iv) True v) True



4.8 MODEL QUESTIONS

1. Define relation and domain in the context of RDBMS.
2. Explain the properties adopted by a Relational Database Management System?
3. What are the advantages of Relational Model?
4. State and explain the basic components of a Relation.
5. Give a brief description of integrity constrains?
6. What is Domain Constraint?

*****

UNIT 5 : KEYS

UNIT STRUCTURE

5.1 Learning Objectives
5.2 Introduction
5.3 Keys
5.4 Types of Keys
  5.4.1 Super Key
  5.4.2 Candidate Key
  5.4.3 Primary Key
  5.4.4 Alternate Key
  5.4.5 Composite Key
  5.4.6 Foreign Key
5.5 Let Us Sum Up
5.6 Further Reading
5.7 Answers to Check Your Progress
5.8 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- learn about the concept of keys and its uses in database
- learn the different types of keys like super key, candidate key, alternate key, primary key, foriegn key etc.
- identify primary and foreign key in a relation
- learn to use composite key

5.2 INTRODUCTION

In our previous unit, we have seen that in case of relational model, the database is logically represented in the form of tables so that it can be easily understood and visualized by everyone.The role of keys are very important in case of relational databases. In fact, without keys relational database will not be usuable at all.

In this unit, we will discuss the concept of keys in a database. The use of different types of keys will be covered in this unit.

5.3 KEYS

In a relational model, a database consists of relations (tables), which consists of tuples (or records/rows), and attributes (or fields/columns). We must have a way to distinguise each tuple within a relation. A relation in a relational database must have an attribute or a combination of attributes such that they can uniquely identify the tuple. This unique identifier is called ***key***. A key is a data item that exclusively identifies a record or tuple. It may consists of one or more attributes. We can split related data into different relations or tables and logically link them together with the help of keys. Without this unique identifier, there is no way to retrieve the unique tuple from a relation.

In this unit, we may use the terminologies *table*, *row* or *record* and *field* in place of *relation*, *tuple* and *attribute* respecively. For example, let us consider the following table “STUDENT”.

Table 5.1: **STUDENT**

Roll_no	Name	Marks	Grade
1	Monirupa Misra	360	A
2	Ranjita Dutta	180	C
3	Rajib Sharma	310	A
4	Kaustab Baruah	265	B
5	Apurba Bora	310	A
6	Rajib Sharma	210	B

Table 5.1 gives us marks and grades of students of a particular class. There are six records in the table “STUDENT”. Each record has the following four fields: ***Roll_no, Name, Marks*** and ***Grade***. As we can see, among the fields *Name*, *Marks* and *Grade*, no one field can identify a record in the table uniquely. The *Name* field, cannot be used as key because several

students might have the same name. *Marks* field may contain same marks for more than one student. Similarly, more than one student may have the the same *Grade*. So these three fields cannot be used as key. However, the field ***Roll_no*** can easily identify any row in the table uniquely. Roll numbers of students in a particular class are different. So such fields can be used as key.

5.4 DIFFERENT TYPES OF KEYS

Every key which has the property of uniqueness can be distinguished as follows :

- Super Key
- Candidate Key
- Primary Key
- Alternate Key
- Composite Key
- Foreign Key

5.4.1 Super Key

A superkey is a set of columns that uniquely identifies every row in a table. For example, if there is a table “STUDENT” with only two columns **Roll_no** and ***Name,*** then the super key will be if we assume that there are no two students in the class with the same Roll_no as well as Name.

{ Roll_no, Name}

Similarly, let us consider an EMPLOYEE table as in Table 5.2 consisting of the columns ***Emp_ID***, ***Name*** and ***Post***. We could use the ***Emp_ID*** in combination with any or all other columns of this table to uniquely identify a row in the table. Examples of superkeys in this table would be ***{Emp_ID}***, ***{Emp_ID, Name}*** and ***{Emp_ID, Name, Post}***.

Table 5.2: **EMPLOYEE**

Emp_ID	Name	Post
001	Goutam Das	Accountant
011	Prakash Bora	Account Asstt.
023	Himangshu Das	Superitendent
033	Niharika Sarma	Financial Officer

In a real database we donot need values for all of those columns to identify a row. We only need a minimal set of columns that can be used to identify a single row. In our example, the set ***{Emp_ID}*** is the minimal superkey.

5.4.2 Candidate Key

A table can have more than one column that could be chosen as the key because they individually have the capability to identify a record uniquely. These fields are termed ***candidate keys***. In other words, a candidate key is any set of one or more columns whose combined values are unique among all occurrences (i.e., tuples or rows or record). Since a NULL value is not guaranteed to be unique, no component of a candidate key is allowed to be NULL. Candidate keys are those attributes of a relation, which have the properties of *uniqueness* and *irreducibility*. Now, let us explain these two properties:

Let K be a set of attributes of relation R. Then K is a candidate key for R if and only if it possesses both of the following properties:

Uniqueness: No legal value of R ever contains two distinct tuples with the same value for K.
Irreducibility: No proper subset of K has the uniqueness property.

Let us consider the following relation ***EMP_INFO*** containing some personal information of employees working in an office. Suppose all of them have passport.

Table: 5.3: **EMP_INFO**

Emp_ID	Name	Passport_no	Blood Group
12341	Kunal Kashyap	M 9523421	A+
12342	Rajib Sharma	M 9515212	O+
12343	Ankur Chakraborty	M 9523123	O+
12344	Niharika Bora	F 9515456	B+
12345	Antara Dutta	F 9643521	AB+

The attribute ***Emp_ID*** and ***Passport_no*** posseses unique data item for each employee. Therefore, any of these two attribute can be chosen as the key. These two are examples of *candidate keys* in Table 5.3. The attribute ***Name*** cannot be a candidate key as more than one employee might have similar name. Similary, several employees might have the same blood group. So ***Blood Group*** cannot be chosen as key.

5.4.3 Primary Key

Every database table should have one or more columns designated as the *primary key*. The value of primary key must be unique for each record in the database. In a database, there can be multiple candidate keys. Out of all the available candidate keys, a database designer decides a primary key. The primary key should be chosen in such a way that its attribute values are never or very rarely changed.

A ***primary key*** is a field or combination of fields that uniquely identify a record in a table, so that an individual record can be located without confusion. Depending on its design, a table or relation may have arbitrarily many unique keys but must have one primary key. For example, let us assume we have a table called ***EMPLOYEE_ADDRESS*** that contains some information for every employee in an organization. We should need to select an appropriate primary key that would uniquely identify each employee. Our first thought might be to use the employee's name i.e, ***Emp_Name***. But this would not work properly because two or more employees with the same

name might be possible in the organization. The ***Location*** field of a person cannot be chosen as primary key since it is likely to change. A better choice might be to use a unique ***Emp_ID*** number that the organization assign to each employee when they have appointed. ***Emp_ID*** can be a primary key as it does not change till the person is working in the same organization.

Table: 5.4: **EMPLOYEE_ADDRESS**

PK

Emp_ID	Emp_Name	Location
1231	Gautam Baruah	GS Road Guwahati
1232	Arindam Dutta	RG Barua Road Guwahati
1233	Meghali Gogoi	Chandmari Guwahati
1234	Bornalee Sharma	Jalukbari Guwahati
1235	Arindam Dutta	Chandmari Guwahati

In the Table 5.1., student's ***Roll_no*** would be a good choice for a primary key. The student's name would not be a good choice, as there is always the chance that more than there may be one student with same name. Some other examples of primary keys are ***Social Security Numbers*** (associated with a specific person) , ***ISBN_no*** (associated with a specific book).

A primary key is a special case of unique keys. *Unique key constraint* is used to prevent the duplication of key values within the rows of a table and allow NULL values. Primary key allows each row in a table to be uniquely identified and ensures that no duplicate rows exist and no NULL values are entered. Thus ***primary key constraint*** can be defined as a rule that says that the primary key fields cannot be NULL and cannot contain duplicate data.

Once we decide upon a primary key and set it up in the database, the database management system (DBMS) will enforce the uniqueness of the key. If we try to insert a record into a table with a primary key that duplicates an existing record, the insert will fail.

Sometimes, a table may have a primary key. In such cases, duplicate records may occur in the table. Most databases are also capable of generating their own primary keys. *Microsoft Access*, for example, may be configured to use the *AutoNumber* data type to assign an unique ID to each record in the table. This is a bad design practice, because it leaves us with a meaningless value in each record in the table. It is better to use that space by storing some useful data.



NULL :
The NULL value indicates that the value does not exist or is not known.

Properties of Primary Key

To qualify as a primary key for an entity, an attribute must have the following properties:



Constraint :
A constraint is a rule that defines what data is valid for a given field.

i) **Stable:**

The value of a primary key must not change or should not become NULL throughout the file of an entity. A stable primary key helps to keep the model stable. For example, if we consider a patient record, the value for the primary key (*Patient number*) must not change with time as would happen with the age field.

ii) **Minimal:**

The primary key should be composed of the minimum number of fields that ensures the occurrences are unique.

iii) **Definitive:**

A value must exist for every record at creation time because an entity occurrence cannot be substantiated unless the primary key value also exists.

iv) **Accessible:**

Anyone who wants to create, read or delete a record must be able to see the primary key value.

5.4.4 Alternate Key

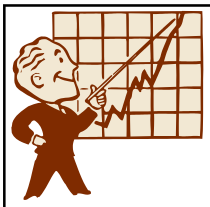
As we have seen, it is possible for a table or relation to have two or more candidate keys. If we choose any one of them as primary key, then the remaining keys will be termerd as alternate key. The alternate

key (or secondary key) is any candidate key which is not selected to be the primary key. For the illustration of alternate key, let us consider the following table ELEMENT which stores some information like element name, symbol, atomic number of the elements of periodic table.

Table: 5.5 : **ELEMENT**

Name	Symbol	Atomic_no
Hydrogen	H	1
Helium	He	2
Lithium	Li	3
Berylium	Be	4
Boron	B	5
Carbon	C	6
Nitrogen	N	7
Oxygen	O	8
Fluorine	F	9
Neon	Ne	10

All the three fields can individually identify each element in the table. So any of these three fields can be chosen as the primary key . If we choose ***Symbol*** as the primary key; ***Name*** and ***Atomic_no*** would then be alternate keys. Similarly, in the ***EMP_INFO*** (Table 5.3), if we consider ***Emp_ID*** as the primary key then ***Passport_no*** will be the alternate key.



CHECK YOUR PROGRESS

1. State whether the following statements are true or false:

   (a) A key is that data item that exclusively identifies a record.

   (b) A table or relation may have arbitrarily many unique keys but at most one primary key.

   (c) The alternate key is any candidate key which is not selected to be the primary key.

   (d) Unique key constraint is used to allow the duplication of key values within the rows of a table and allow NULL values.

   (e) The primary key fields cannot be NULL and cannot contain duplicate data.

5.4.5 Composite Key

In some situations, while designing a database, there may not be a particular column or field that can individually identify a record uniquely in a table. In such cases, we may require to select two or more fields so that combination of those can identify each record uniquely. This combination of fields is known as ***composite key***. It is used when a record cannot be uniquely identified by a single field.

For the illustration of composite key, let us consider Table 5.6. ***ITEM*** with the fields ***Supplier_ID, Item_ID, Item_Name*** and ***Quantity***. This table gives us the information that a supplier supply some items. As we can see, any of these fields indivisually cannot identify a row in the table uniquely. But if we combine ***Supplier_ID*** and ***Item_ID***, then these together can easily identify any row in the table uniquely. Thus, ***Supplier_ID*** and ***Item_ID*** together becomes a composite key.

Table: 5.6 : **ITEM**

Composite Key

Supplier_ID	Item_ID	Item_Name	Quantity
S_1	I_1	AC	5
S_1	I_2	Inverter	8
S_2	I_2	Inverter	4
S_2	I_3	UPS	15
S_2	I_4	Generator	5
S_3	I_3	UPS	10



Entity Integrity: *According to entity integrity, no key column of any row in a table can have a NULL value. In other words, if an attribute of a table is prime attribute (unique identifier), it cannot accept NULL values.*



Referential Integrity: *To ensure that a value which appears in one relation or table for a given set of attributes will also appear for certain set of attributes in another relation. This condition is known as referential integrity.*

5.4.6 Foreign Keys

In this section we will discuss the concept of ***foreign key***. These keys are used to create relationships between tables.

A foreign key is a field in one relational table that matches the primary key column of another table. It identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table. The columns in the referencing table must be the primary key or other candidate key in the referenced table. The values in one row of the referencing columns must occur in a single row in the referenced table. Thus, a row in the referencing table cannot contain values that donot exist in the referenced table. This way references can be made to link information together and it is an essential part of database normalization. Multiple rows in the referencing table may refer to the same row in the referenced table.

For example in an employees database, let us imagine that we wanted to add a table ***DEPARTMENT*** containing departmental information to the database. We would also want to include information about the employees in the department, but it would be redundant to have the same information in two tables (***EMPLOYEE*** and ***DEPARTMENT***). Instead, we can create a relationship between the two tables. Let us assume that the ***DEPARTMENT*** table uses the ***Department_Name***

column as the primary key. To create a relationship between the two tables, we add a new column to the ***EMPLOYEE*** table called ***Department_Name***. We then fill in the name of the department to which each employee belongs. The ***Department_Name*** column in the ***EMPLOYEE*** table is a *foreign key (FK)* that references the ***DEPARTMENT*** table. The database will then enforce *referential integrity* by ensuring that all of the values in the ***Department*** column of the ***EMPLOYEES*** table have corresponding entries in the ***DEPARTMENT*** table.

Table: 5.7: **EMPLOYEE**

PK(Primary key) FK(Foreign key)

Emp_ID	**Name**	**Post**	**Department_Name**
A01	Goutam Bora	Accountant	Sales
A02	Manash Saikia	Manager	Marketing
A03	Himangshu Das	Financial Officer	Human Resource
A04	Niharika Baruah	Accountant	Production
A05	Pranjal Hazarika	Law Officer	Human Resource
A06	Meghna Roy	Receptionist	Executive Sales
A07	Diganata Bora	Manager	Production
A08	Smita Roy	Law Officer	Production
A09	Barnalee Sharma	Receptionist	Marketing
A10	Dhrub Sharma	Manager	Human Resource
A11	Kamalesh Das	Manager	Sales

Table: 5.8: **DEPARTMENT**

PK

Department_Name	**Manager**	**Dept_Code**
Human Resource	Dhruba Sharma	D1
Marketing	Manash Saikia	D2
Production	Diganata Bora	D3
Sales	Kamalesh Das	D4

Again, let us consider a book database. The ***BOOKS*** table has a link to the ***PUBLISHERS*** table. The ***Pub_ID*** column is the primary key for the ***PUBLISHERS*** table and ***ISBN_no*** is the primary key for the ***BOOKS*** table. The ***BOOKS*** table also contains a ***Pub_ID*** column which matches the primary key column of the ***PUBLISHERS*** table. This ***Pub_ID*** is the foreign key in the ***BOOKS*** table. The ***Pub_ID*** field in the ***BOOKS*** table indicates which publisher a book belongs to.

Table: 5.9: **PUBLISHERS**

PK

Pub_ID	Pub_Name	City	State

Table:5.10: **BOOKS**

PK FK

ISBN_no	Book_name	Author_name	Pub_ID	Price	Pub_date

Although the primary purpose of a ***foreign key constraint*** is to control the data that can be stored in the foreign key table, it also controls changes to data in the primary key table. For example, if the row for a publisher is deleted from the ***PUBLISHERS*** table, and the publisher's ID is used for books in the ***BOOKS*** table, the relational integrity between the two tables is broken; the deleted publisher's books are orphaned in the ***BOOKS*** table without a link to the data in the ***PUBLISHERS*** table. A foreign key constraint prevents this situation. The constraint enforces referential integrity by ensuring that changes cannot be made to data in the primary key table if those changes invalidate the link to data in the foreign key table. If an attempt is made to delete the row in a primary key table or to change a primary key value, the action will fail if the deleted or changed primary key value corresponds to a value in the foriegn key constraint of another table. To change or delete a row in a foreign key constraint successfully, we must first either delete the foreign key data in the foreign key table or change the foreign key data in the foreign key table, thereby linking the

foreign key to different primary key data. i.e., a primary key constraint cannot be deleted if referenced by a foreign key constraint in another table; the foreign key constraint must be deleted first.



CHECK YOUR PROGRESS

2. State whether the following statements are True or False:

   (a) In a relational database, the foreign key of a relation would be the primary key of an another relation.

   (b) Foreign Key represents relationship between the tables.

   (c) There may be more than one foreign key in a table.



5.5 LET US SUM UP

- A key is that data item that exclusively identifies a record. A database consists of tables, which consists of records, which further consist of fields.
- Database keys can be classified into *super key*, *candidate key, primary key, alternate key, composite key* and *foreign key*.
- A candidate key is any set of one or more columns whose combined values are unique among all occurrences (i.e., tuples or rows).
- A super key for an entity is a set of one or more attributes whose combined value uniquely identifies the entity in the entity set. A super key has the uniqueness property but not necessarily the irreducibility property .
- A primary key is a special case of unique keys. The major difference is that for unique keys the NOT NULL constraint is not automatically enforced, while for primary keys it is. Thus, the values in a unique key columns may or may not be NULL.
- The primary key of any table is any candidate key of that table which

the database designer arbitrarily designates as “primary”. The primary key may be selected for convenience, comprehension, performance, or any other reasons.

- The alternate key is any candidate key which is not selected to be the primary key.
- A composite key is made up of two or more columns.
- Foreign keys are very important in the context of table inter-relationships. It is a field in a relational table that matches the primary key column of another table.



5.6 FURTHER READING

1. Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems. Pearson.
2. Date, C. J. (2006). An introduction to database systems. Pearson Education India.
3. Singh, S. K. (2011). Database systems: Concepts, design and applications. Pearson Education India.



5.7 ANSWERS TO CHECK YOUR PROGRESS

1. (a) True (b)True (c) True (d) False (e) True
2. (a) True (b) True (c) False



5.8 MODEL QUESTIONS

1. Explain the following keys with examples:
   (i) Primary Key
   (ii) Candidate Key
   (iii) Super Key

(iv) Alternate Key

(v) Foreign Key.

2. Create a relation student and mark the different types of keys in it.
3. Explain the significance of primary key and foreign key in employee database.
4. Define key?
5. What is the difference between candidate key and super key? Explain.
6. Discuss the properties of primary key?
7. Define candidate key with appropriate example?
8. Define primary key constraint.
9. Define the uniqueness and irreducibility properties of candidate key.
10. Design two database table showing primary key and foreign key.

*****

UNIT 6 : RELATIONAL DATABASE DESIGN

UNIT STRUCTURE

6.1 Learning Objectives
6.2 Introduction
6.3 Design Guidelines
6.4 Anomalies in a Database
6.5 Functional Dependency
  6.5.1 Full Functional Dependency
  6.5.2 Partial Dependency
  6.5.3 Transitive Dependency
  6.5.4 Multi-Valued Dependency
  6.5.5 Join Dependency
6.6 Decomposition
  6.6.1 Decomposition and Dependency Preservation
  6.6.2 Decomposition and Lossless Join
6.7 Prime and Non Prime Attribute
6.8 Universal Relation
6.9 Let Us Sum Up
6.10 Further Reading
6.11 Answer to Check Your Progress
6.12 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn how to design a good database
- learn about functional dependency
- discuss different types of anomalies in a database
- define the decomposition nature of a database

6.2 INTRODUCTION

We are familiar with the concept of database and DBMS from the previous units. We have acquainted with relational model and their constraints and keys.

This unit focuses on relational database design. The main goal of relational database design is to generate a set of schemas that allow us to store data without unnecessary redundancy as well as retrieve information easily and accurately. Here, we will discuss some important concept like anomalies in a database, functional dependency, decomposition etc. which are related with database design process.

6.3 DESIGN GUIDELINES

Designing a database is a very important task. If we design a database correctly, storing, updating and retrieving of data will be efficient. Some important features of a good database design are as follows:

Semantics of the Relation Attributes :

Semantics specifies how the attribute values in a tuple are related to one another. To explain the semantics, it is better to design relation schema. So, we have to design a relation schema in such a way that it is easy to explain its meaning. We should not combine the attributes from multiple entry types and relationship types into a single relation.

Reducing the redundant values in tuples :

One goal of schema design is to minimize the storage space. Grouping of attributes into relation schemas has a significant effect on storage space. We have to design the base relation schemas so that no insertion, deletion and modification anomalies occur in the relations.

Null values in tuples:

A null value in a relation is the wasteful of space at the storage level that may lead problems in understanding the meaning of the attributes. So it is better to avoid null values in base relation. If nulls are unavoidable, it is necessary to make sure that they apply in exceptional cases only and do not apply to a majority of tuples in a relation.

Disallowing spurious tuples:

We should design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples are generated.

6.4 ANOMALIES IN A DATABASE

The aim of the database system is to reduce redundancy, i.e., unnecessary repetation of data should be eliminated. Updates to the database with such redundancies have the potential of becoming inconsistent. The inconsistencies are explained below.

Update anomalies:

Multiple copies of the same fact may lead to update anomalies or inconsistencies when an update is made and only some of the multiple copies are updated. Consider the STUDENT relation of Figure 6.1.

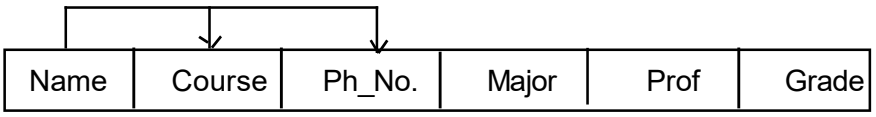


Fig. 6.1: Functional dependencies of STUDENT relation

In the above figure 6.1, it is assumed that one student can be enrolled for more than one courses. Here we have considered that there are three entries with the name “Rahul” with different courses. The phone numbers of three tuples are same because the entries are for one student Rahul. Suppose Rahul has taken a new phone number. Now a change in the phone number (Ph_No.) of Rahul must be made in all tuples pertaining to the student Rahul for consistency. If one of these three tuples is not changed to reflect the new phone no. of Rahul, there will be an inconsistency in the data.

Insertion anomalies:

If above is the only relation in the database showing the association between

a faculty member and the course he or she teaches, the fact that a given professor is teaching a given course cannot be entered in the database unless a student is registered in the course. Also, if another relation also establishes a relationship between a course and a professor who teaches that course the information stored in these relations has to be consistent.

Deletion Anomalies:

If the only student registered in a given course discontinues the course, the information as to which professor is offering the course will be lost if this is the only relation in the database showing the association between a faculty member and the course she or he teaches. If another relation in the database also establishes the relationship between a course and a professor who teaches that course, the deletion of the last tuple in STUDENT for a given course will not cause the information about the course's teacher to be lost.



CHECK YOUR PROGRESS

1. Fill in the blanks.

i) Redundancy leads to _____.

ii) Avoiding redundancy is important because it _____.

iii) Faithfulness is one _____ of a good database design.

iv) Multiple copies of the same fact may lead to _____ anomalies.

v) A null value in a relation is the wasteful of space at the _____.

vi) It is always better to avoid null values in _____.

6.5 FUNCTIONAL DEPENDENCY

A ***functional dependency*** is a constraint between two sets of attributes from a database. A functional dependency denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of relation R. It specifies a constraint on the possible tuples that can form a relation instance r of R.

The constraint states that for any two tuples t1 and t2 in r such that t1[X] = t2[Y], it must also have t1[X] = t2[Y]. This means that values of the X component of a tuple uniquely or functionally determine the values of the Y component.

For example, let us consider the following **ITEM** table, shown in Table 6.1. Let us consider a combination of the ***Item_code*** and the ***Item_name*** columns. Item_code is the primary key of the table and therefore always unique. If Item_code given, we can determine Item_name. Given the value of *Item_code*, there is only one value of *Item _name* for it. Thus, *Item_name* is functionally dependent on *Item_code*. Symbolically, this is written as:

Item_code → Item_name

This should be read as ***Item_code*** functionally determines ***Item_name***.

Table 6.1: ITEM table

ITEM

Item_code	Item_name	Quantity
P01	Pen	200
P02	Marker	150
P03	Pencil	300
P04	Eraser	250

Functional dependency may also be based on a composite attribute. For example, if we write

{X,Y} → Z

It means that Z is functionally dependent on composite attribute X and Y.

A functional dependency is a property of the meaning or semantics of the attributes in a relation schema R. If we consider the Figure 6.1, the functional dependencies are:

Name → {Ph.No., Major}
Course → Prof
{Name, Course} → Grade

Here, in the first relation, the attributes ***Ph.No*** and ***Major*** are functionally dependent on the prime attribute ***Name***. Alternatively, we can say that the prime attribute ***Name*** determined the non prime attributes ***Ph.No*** and ***Major***. Similarly, we can now explain the other two relational schemas also. Concept of prime and non prime attribute will be discussed at the end of this unit.

6.5.1 Full Functional Dependency

When all non-key attributes are dependent on the key attribute, it is called full functional dependency. For example, consider the relation schema of an employee,

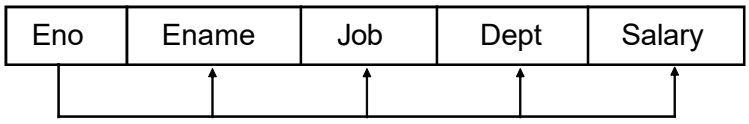


Fig. 6.2: Functional dependency of relation EMPLOYEE

Here all non-key attributes (Ename, Job, Dept, Salary) are dependent on key attribute Eno, which is the Employee No. in the relation EMPLOYEE.

6.5.2 Partial Dependency

A functional dependency X $\rightarrow$ Y is a partial dependency if some attributes A $\rightarrow$ X can be removed from X and the dependency still holds. It means that all non-key attributes are not dependent on the key attribute. There is a partial dependency of non-key attributes either on the key attribute or on the non-key attribute.

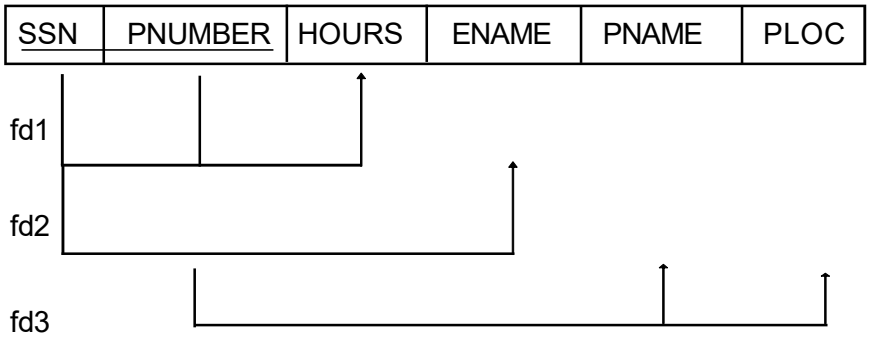


Fig. 6.3: A relation schema of Emp-Project relation

In figure 6.3, the dependency
{SSN, PNUMBER} → ENAME is partial because SSN → ENAME holds.

6.5.3 Transitive Dependency

A functional dependency X → Y in a relation R is a transitive dependency if there is a set of attributes Z that is not a subset of any key of R and both X → Z and Z → Y hold. In general sense we can say that if A, B, C are three attributes in a table, and if A is related to B and B is related to C, then A is indirectly related to C.

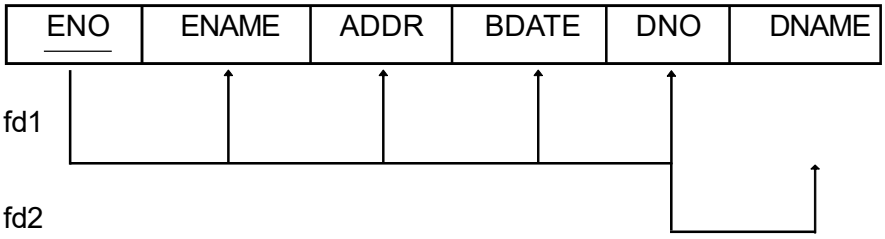


Fig. 6.4: A relation schema of Emp-Dept relation

In the above figure, DNO is dependent on key attribute ENO and DNAME is dependent on DNO which we can denote as,

ENO → {ENAME, ADDR, BDATE, DNO}
DNO → DNAME

From the above dependencies we can say that DNAME is indirectly related to key attribute ENO. So, DNAME is transitively dependent on ENO.

6.5.4 Multi-Valued Dependency

Let R be a relational variable and A, B and C are subsets of the attributes of R. Then we say that B is multi-dependent on A. Symbolically, it can be denoted as,

A $\rightarrow\rightarrow$ B and read as A multi-determines B

Multi-valued dependencies are a generalization of functional dependencies, in the sense that every functional dependency is an multi-valued dependency. But the converse is not true.

COURSE	TEACHER	TEXT
Data structure	Prof. Baruah	Basic data structure
Data structure	Prof. Sarma	Data structure through C
Digital Logic	Prof. Reddy	Principle of Logic gates
Digital Logic	Prof. Talukdar	Basic logic structure
Digital Logic	Prof. Sarma	Principle of Digital Circuit
Digital Logic	Prof. Baruah	Fundamental of Logic gates

Fig. 6.5 : A relation table of relation COURSE

In the above table there are two multi-valued dependencies that hold:

COURSE $\rightarrow\rightarrow$ TEACHER

COURSE $\rightarrow\rightarrow$ TEXT

The first multi-valued dependency means that although a course does not have a single corresponding teacher, each course does have a well-defined set of corresponding teaches. Multi-valued dependency does not rule out the existence of certain tuples, which have multiple dependencies. Instead, they require that other tuples of a certain form be present in the relation.

6.5.5 Join Dependency

Join dependency is a constraint, similar to a Functional dependency or a Multi-valued dependency. It is satisfies if and only if the relation concerned is the join of certain number of projections. The definition is if R be a relational variable, and A, B, ..., Z are subsets of attributes

of R, then R satisfies the Join Dependency that is denoted a,

* {A, B, ..., Z} and read as star A, B, ..., Z

6.6 DECOMPOSITION

The decomposition of a relation scheme R = (A_1, A_2, ..., A_n) is its replacement by a set of relation schemas { R_1, R_2, ..., R_m}, such that $R_1 \leq R$ for $1 \leq i \leq m$ and $R_1 \cup R_2 \cup R_m = R$. A relation schema R can be decomposed into a collection of relation schemas {R_1, R_2, ..., R_m} = D to eliminate some of the anomalies contained in the original relation R. Here the relation schemas $R_1 (1 \leq i \leq m)$ are subsets of R and intersection of $R1 \cap Rj$ for $i \neq j$ need not be empty. Furthermore the union of Rj ($1 \leq i \leq m$) is equal to R, i.e $R = R_1 R_2 ... R_m$.

6.6.1 Decomposition and Dependency Preservation

We need to preserve the dependencies because each dependency in functional dependency F represents a constraint on the database. If one of the dependencies is not represented by the dependencies on some individual relation R_i of the decomposition, it will not be able to enforce this constraint by looking only at an individual relation. Instead to enforce the constraint, we will have to join two or more of the relations in the decomposition and then check that the functional dependency holds in the result of the join operation. It is useful that each functional dependency $X \rightarrow Y$ specified in F either appear directly in one of the relation schemas Ri in the decomposition D.

We say that a decomposition D = {R1, R2, ..., Rm} of R is dependency preserving with respect to F if the union of the projections of F on each Ri in D is equivalent to Fj, that is

$$((\pi_F (R_1)) \ ... ... \ (\pi_F (R_m)))^+ = F^+$$

If any decomposition is not dependency preserving, some dependency is lost in the decomposition. We can check by joining several relations in the decomposition. Let us consider the relation of STUDENT with their functional dependencies.

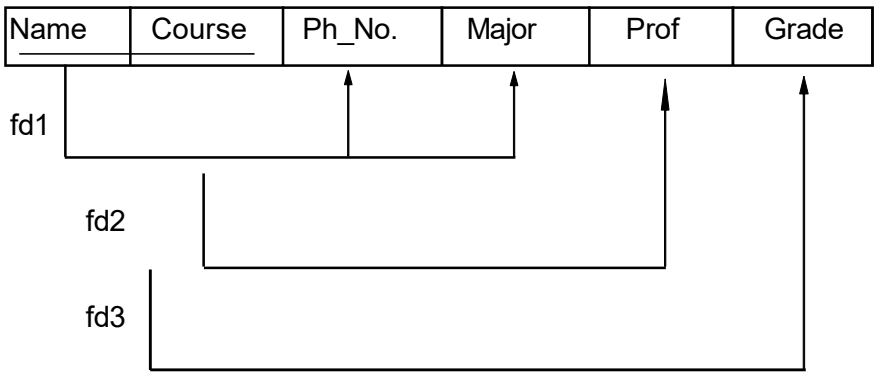


Fig. 6.6 Functional dependency of relation STUDENT

The dependency will resolve if we decompose the above relation with the following relations.

STD_INFO (Name, Ph_no, Major)
STD_RES (Name, Course, Grade)
TEACHER (Course, Prof)

The **first relation** schema gives the phone number and major of each student and such information will be stored only once for each student. Any change in the phone number will thus require a change in only one tuple of this relation. The **second relation** schema stores the grade of each student in each course that the student is or was controlled in. The **third relation** schema records the teacher of each course.

6.6.2 Decomposition and Lossless Join

This property ensures that no spurious tuples are generated when a **natural join** operation is applied to the relations in the decomposition.

It is the property of a decomposition of relation schemas that no spurious tuples should hold on every legal relation instances. The lossless join property always defined with respect to a specific set F of dependencies. The word “lossless” refers to loss of information, not loss of tuples.

Formally, a decomposition D = {R1, R2, ... ,Rm} of R has the lossless join property with respect to the set of dependencies F on R if for every relation instance r of R that satisfies F, the following holds,

$$*(\pi_{<R1>}(r),.....,\pi_{<Rm>}(r)) = r$$

If decomposition does not have lossless join property, then it may get an additional spurious tuples after the projection or natural join operation applied. These additional tuples represent erroneous information and hence add wrong information. It is also called *non-additive join* because it describes the situation more accurately.



CHECK YOUR PROGRESS

2. Fill in the blanks

a) When all non key attributes are dependent on the key attribute, it is called _____ dependency.

b) When one non key attribute depends on other non key attribute, it is called a _____ dependency.

c) A functional dependency is denoted by, _____ between two sets of attributes A, B.

d) Multi-valued dependencies are also called as _____.

e) A multi-valued dependency is denoted by, _____ between two sets of attributes A,B.

f) The _____ of a relation scheme is its replacement by a set of relation schemas.

g) A relation schema can be decomposed into a collection of relation schemas to eliminate some of the anomalies contained in the original relation.

h)The _____ property guarantees the spurious tuple problems.

i) The lossless join property is also known as _____.

j) A _____ attribute is a member of any key of a relational schema.

3. Write whether the following statements are true or false:

a) A functional dependency is a constraint between two sets of attributes from the database.

b) The lossless join property cannot give the guarantee of the spurious tuple problem.

c) Dependency preservation property ensures that all FDs are represented in some of the individual resulting relations.

d) Join dependency is a join of certain number of projection of relation schemas.

e) Multi-valued dependency is also referred as equality generating dependency.

6.7 PRIME AND NON PRIME ATTRIBUTE

An attribute of relation schema R is called a prime attribute of R if it is a member of any key of R. An attribute is called non prime attribute, if it is not a prime attribute, i.e. it is not a member of any candidate key of R.

SSN	PNUMBER	HOURS

Fig. 6.7 : a relation schema of relation WORKS_ON

In the figure both SSN and PNUMBER are prime attributes and HOURS is non prime attribute.

6.8 UNIVERSAL RELATION

The concept of the universal relation manifests itself in several different ways. First of all it includes all of the attributes relevant to the database

under consideration and then showed how those relational variables can be replaced by successively smaller projections until some good structure is reached.

The second, and more pragmatically significant, manifestation of the universal relation is the ***user interface***. The basic idea here is quite straightforward, and indeed quite appealing: Users should be able to frame their database requests, not in terms of relational variables and joins among those relational variables, but rather in terms of attributes alone.



CHECK YOUR PROGRESS

4. Differentiate prime and non prime attributes for the following STUDENT relation.

RNO	SNAME	MARKS	DNO	DNAME

fd1

fd2

6.9 LET US SUM UP

- The design and implementation of a database relation should be faithful to the requirements. Simplicity requires that the design and implementation avoid introducing more elements than are absolutely necessary – Keep it Simple.
- Attributes are easier to implement, but entity sets and relationships are necessary to ensure that the right kind of element is introduced.
- The aim of the database system is to reduce redundancy, meaning that information is to be stored only once. Storing information several times leads to the waste of storage space and an increase in the total size of the data stored. Updates to the database with such redundancies have the potential of becoming inconsistent.
- Functional Dependencies are the consequence of the interrelation-

ships among attributes of an entity represented by a relation. It is denoted by X $\rightarrow$ Y, between two sets of attributes X and Y.

- When all non-key attributes are dependent on key attribute, it is called full functional dependency.
- In partial dependency, all non-key attributes are not dependent on the key attribute.
- Functional Dependencies rule out certain tuples from being in a relation, but multi-valued dependency do not rule out the existence of certain tuples, which have multiple dependencies. Functional dependencies are ***equality generating dependencies*** and Multi-valued dependencies are ***tuple generating dependencies***.
- The reduction process of a relation consists of replacing the given relational variable by certain projections, in such a way that joining those projections back together again gives us back the original relational variable. The process is reversible.
- Decomposing a universal relation schema satisfies the lossless join property, dependency preservation property or both properties. All these properties are based on the functional dependencies specified on the attributes of the universal relation.



6.10 FURTHER READING

1.Elmasri, R., & Navathe, S. B. (2015). Fundamentals of database systems. Pearson.

2. Date, C. J. (2006). An introduction to database systems. Pearson Education India.

3.Singh, S. K. (2011). Database systems: Concepts, design and applications. Pearson Education India.

6.11 ANSWER TO CHECK YOUR PROGRESS

1. i) Inconsistency ii) Waste space iii) Feature
   iv) Update v) Storage level vi) Base relation

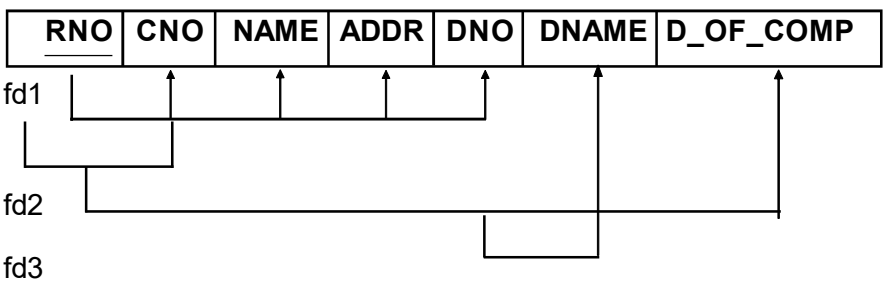
2. (a) full functional dependency (b) transitive dependency
   (c) A → B (d) tuple generating dependency
   (e) A →→ B (f) decomposition
   (g) anomalies (h) lossless join
   (i) non adaptive join (j) prime

3. (a) True (b) False (c) True (d) True (e) False

4. Prime: RNO, DNO
   Non prime: SNAME, MARKS, DNAME

6.12 MODEL QUESTIONS

1. Define Functional Dependency? Why it is required?
2. Explain Partial Dependency with a suitable example.
3. What is Multi-valued and Join Dependency?
4. Consider the following relational schema of STUDENT relation where RNO is the Roll no of student, CNO is the Course Enrolment no for the student and DNO is the Department no. Relation shows their functional dependencies also. Decompose it, so that it preserves the dependencies.



5. Write short notes on:
   a) Transitive dependency
   b) Multi-valued Dependency

c) Decomposition

d) Universal Relation

e) Dependency Preservation

f) Decomposition with Lossless Join

6. Differentiate Prime attribute and Non-prime attribute with example.
7. State the features of a good database design
8. Explain the inconsistency arise at the designing time of databases.

****